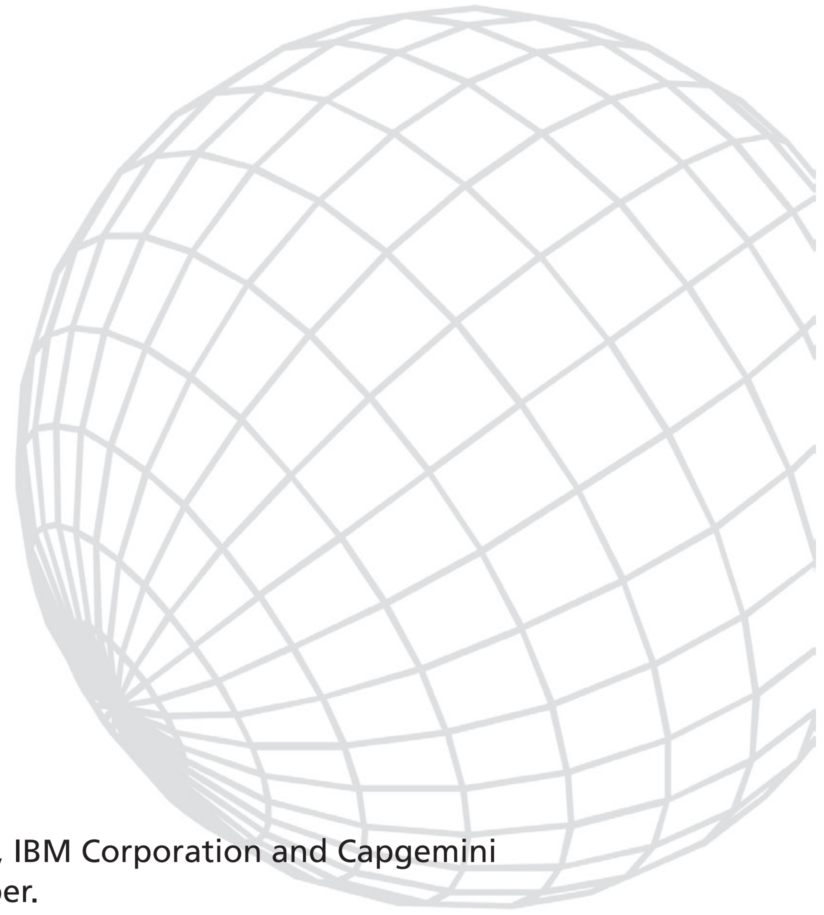
SOA in Manufacturing Guidebook



WHITE PAPER 27
A MESA International, IBM Corporation and Capgemini co-branded white paper.
05.21.08

Table of Contents

List of Figures

INTRODUCTION

Manufacturing companies are facing many new challenges today to become more flexible and agile as business models change. Companies' ability to adapt quickly to a changing business environment mainly depends on the agility of their corporate cultures, flexibility of their business processes and interoperability of their IT system(s) they employ. Unfortunately, many manufacturing companies today have IT systems that are inflexible, antiquated, and are difficult and expensive to enhance, maintain and support.

One driver of business change is the increasing number of regulations that government and other agencies have imposed on manufacturing companies. Life sciences, food and beverage, aerospace and defense, and automotive companies are all required to track, trace, and create genealogy in the manufacturing of their products from so called "cradle to grave", i.e. from raw materials to finished goods. In some cases, the company must trace not only their main product produced through its entire life cycle with end customers but also their secondary or incidental products deriving from the core manufacturing process.

Another business shift, or trend, requiring flexibility today is the use of many different suppliers to manufacture the end product. Increasingly, OEMs are using more pre-assembled parts from a globally distributed network of suppliers who come and go rather quickly. In order to maintain profitability, companies need to seamlessly and securely integrate their IT systems to suppliers' in order to track product, supplies, schedules, etc. The IT systems of both OEM and supplier need to be flexible enough to handle different requirements as different suppliers and OEMs do business.

One technology or architecture that helps companies with this problem is called Services Oriented Architecture, or SOA. SOA, used in combination with appropriate industry standards and continuous improvement (CI) methods, allows for a plug-and-play type of architecture for IT systems. In essence the IT system's functionality can be added, changed or removed quickly as market demands require business changes.

This paper discusses the current business drivers and trends in manufacturing industries, and explores how those drivers and trends are causing companies to re-think their IT architecture. The paper introduces SOA and its different components. It also discusses the new tools that are available to help companies realize the benefits of SOA. There are different means to accomplish SOA depending on the technology and development platform chosen. Two of the popular approaches are Microsoft® Windows®[1] Communication Foundation (WCF) and the Java 2 Enterprise Edition (J2EE) framework. While there are many similarities in these approaches, there are some differences. This paper does not attempt to draw distinctions. Given the expertise and experience of the authors, this paper is written from the J2EE perspective using the generally accepted standards of that industry.

BUSINESS DRIVERS AND TRENDS
Key Manufacturing Trends

Lean Manufacturing

Manufacturing trends come and go in ever evolving industrial manufacturing industry. They arise suddenly in the form of a new technical or social development that manufacturing companies respond to by creating corporate initiatives. At some point either the trend goes main stream and becomes "business as usual" or it is displaced by other initiatives. There seems to be one exception; most manufacturers, under ongoing competitive pressure to continuously improve, are utilizing lean manufacturing. This is especially true in the automotive industry, since lean manufacturing arose there first as the Toyota Production System.

"Lean Manufacturing" methodologies are a major part of continuous improvement business model. General Motors and Ford have their version of the Toyota Production System. Other industries are applying it as well. Lean has to do with reducing waste in order to improve all key measures of manufacturing performance: quality, asset utilization, safety, materials management, cost, delivery. In Lean manufacturing, every person in the plant and business is a decision maker thus driving decision making authority to the area where the work is being preformed. Each person learns to see the forming of workflow bottlenecks and waste and then knows the process to correct the condition quickly.

Lean is not just inventory or cost reduction as many America companies perceive it; Lean is about creating a flexible combination of organization and systems to adapt workflows of manufacturing and supply chains to the instantaneous demand requirement of the market. This means a company must be able to identify the demand state and reconfigure their resources to address bottlenecks and waste to optimize each work order path to achieve the highest profitable performance for that work order. The business benefits of a well implemented Lean manufacturing program have been well demonstrated and documented. All are challenged to continuously improve based on market demand. Otherwise, the gains from an initial implementation erode over time because they no longer align with the demand state. Another challenge is consistency with management turnover. Consumers expect consistent quality across the product line. A failure to perform and deliver quality from one plant to another for one product usually results in the customer leaving the brand entirely and switching to a different provider. It's not good enough for one plant to be world class—they all have to be.

The best manufacturers apply Lean on the shop floor and extend it elsewhere in the company, to practice Lean warehousing, Lean logistics, etc. Some Lean manufacturing techniques, like value stream mapping, are applied throughout the enterprise. The best manufacturers continuously improve their performance. Lean manufacturing never becomes business as usual. Process improvement is a continuous and a dynamic business initiative because it is dictated by market demands, technology changes and scaling of new product introductions. There's always room for improvement.

Six Sigma in Manufacturing

Six Sigma is a methodology for statistical analysis with a goal of minimizing process variation. Six Sigma might have a more typical life cycle for CI method. It's easy to misapply. Lessons learned from Six Sigma failures may lead to more careful use of the techniques in the context of Lean manufacturing.

One reason why Six Sigma is sometimes deployed top-down across the enterprise is to help companies develop a culture of data-driven decision making. In Six Sigma training, this is drilled into the participants, and everyone in the company who takes the training. The natural tendency is to make decisions based on emotion and politics. Sometimes decisions are made on the basis of experience, but years of experience can mean that the wrong habits have been ingrained and sometimes legacy standard operating practices incorporate these bad behaviors

3M successfully deployed Six Sigma through: 1) a shared language, 2) a culture of data-based decision making, and 3) breaking down departmental silos. The CEO drove it by making it "not optional" where he said to everyone, "We will make data-based decisions." Conversely, others say that Six-Sigma nearly drove out all of 3M's world-leading capability to innovate. Why?

Getting a culture of data-driven decision making is a key change for a company. But the value can be dulled if the Six Sigma program is focused too much on the tools and on slavishly following the process steps. You have to know when to apply the statistical tools. If the facts and root causes are clear without applying all the Six Sigma steps, tools, bureaucracy, and weeks and months of work, then proceed directly with the facts at hand. Six Sigma results are also dulled if the projects are too focused on financial and sales processes and not on engineering, supply, manufacturing operations, delivery and support processes. Many times, results from different processes directly contradict one another and must be balanced. This is what Toyota recognized by requiring cross-departmental teams for all CI projects.

A recent blog from www.iSixSigma.com: "The Six Sigma methodology has been misapplied by check-sheet commandos and quant jocks that can't deviate from their Six Sigma roadmap."

Whether applied top-down across the entire enterprise, or applied in limited context, Six Sigma and data-driven decision making create demands on IT for more reliable and correct measurement systems and information. Specialized software to support statistical analysis and other Six Sigma and more broadly based Lean manufacturing techniques are also in demand.

Manufacturing IT in Plants

A more recent trend, over the past ten to fifteen years, is the widespread and pervasive deployment of computing technology into plants. With powerful servers and a variety of software packages becoming lower cost to acquire and deploy, manufacturers have implemented many more applications than in the early 1990s. Some plants now use hundreds of different applications, with some running on devices or embedded in manufacturing equipment on the shop floor, and some hosted at the plant's local internal data center near the plant management offices.

However, this first age of manufacturing IT was based on disparate applications comprised of different data models, application architecture, transactions, and messaging constructs. The cost of ownership of these applications is very high due to lack of similarity, flexibility, and integration methods. Consequently, many plants utilize spreadsheets and word documents for data collection and analysis and for other information processing, in addition to the plant applications that are available to them. This makes data warehousing, correlation, analysis and event-driven workflows impossible to do in a cost effective manner. According to AMR Research, this need for integrated manufacturing IT architectures to support global manufacturing is illustrated in corporate IT budget for manufacturing operations increases from 3% in 2001 to 19% in 2007.

Manufacturing IT strategy must be driven by a continuous improvement business strategy. IT and manufacturing departments must implement a company's strategy throughout their manufacturing strategies and systems. At the best manufacturers, IT's focus is on supporting and enabling improvements in work practices. IT architectures and systems need to continue to identify and improve manufacturing metrics to represent the current manufacturing state of change. In automotive, this would include product launch timing for new powertrains and vehicles, initial quality, warranty and scrap reduction, asset utilization, and delivery to schedule.

To support Lean manufacturing, the primary IT responsibility is to ensure the right information is available when needed by decision makers (which is every person in the plant, remember) to make correct, timely decisions. Some smart, automated devices are beginning to be programmed to identify, analyze and correct workflow as well, especially in high volume, highly automated facilities. Business processes are starting to be embedded in IT systems. SOA accelerates the Lean IT approach. To make an improvement in work practice permanent, changes to the IT systems are needed. Over time, positive changes accumulate and compound to provide substantial, sustained business benefits.

Trends Leveraged in Combination Deliver Results

At ARC Advisory Group's manufacturing conferences, manufacturing executives report results delivered by leveraging CI trends of improved IT, Lean manufacturing, great communication and involvement with people, enterprise standards, Six Sigma, and better information. In Table 1, IT Enabled Continuous Improvement Examples, are examples driven by combining IT and CI methods to make manufacturing more adaptable for the global market.

Table 1: IT Enabled Continuous Improvement Examples

Approach	**Results**
Redesigned work processes, which uncovered that IT architectures are fragmented. Put in place a common platform: single instance of ERP package. Factors in productivity improvement: automation, greater efficiency of human capital, better organizational design, improved tools in IT space. "It's change management from the top: a leadership model that absolutely, totally believes in the power of the people around you. You've got to communicate, you've got to engage, and you've got to have goals and metrics against the strategy. These sorts of things aren't specific to IT, but IT is central to many of them. Before we can even begin to address the IT infrastructure, we need to understand the strategic plan for the company."	80% reduction in response time per transaction while volume of transactions grew 450%. Since 1996 completed more than 2000 IT projects delivering measured value of $2.5 billion since 2006. Total IT cost flat (1997–2004). 1992–2004 productivity up 8% per year.
Globally common software (single family of PLCs), common hardware (standard panels), standard open communication networks, training, system designs, simulation applications that leverage the commonality. Architecture for controls applications that scales to small as well as large plants.	Savings of 50–70% in acquiring controls equipment. Reduced costs of deploying and supporting plant applications globally.
Integrated IT strategy that led to using Manufacturing Execution System (MES) to enforce workflow and provide information on where to improve. Must start as an integrated IT strategy. Lasting benefit has come from creating a data-driven culture focused on continuous improvement. It's hard to hold someone accountable with bad data.	Reduced cycle time to turn around order from 4 weeks with gross margin -4% in 2001 to 5 days in 2006 with gross margin 69%. On time delivery has been 100% for years. Cost per order reduced 25%.
In a Lean system, stability controls the speed of improvement. Manufacturing Execution System is vital to process stability. Using the MES you monitor yield data, process parameter data, defect detection, defect mapping, build data management.	Deployed Lean and Six Sigma in a way that received full support from the plant managers. Manufacturing Execution Systems provide the visibility they need. Results at one plant: productivity increased 60%, reduced inventory 50%. Another plant: productivity up 40%.
Lean: Line layout design, material flow, information flow, change of incentives and work content. Don't limit yourself to the original design intent of IT system functionality, and don't use software functionality just because it's there. Use just what helps you move the business forward.	Strong concurrent implementation of newly purchased ERP system while implementing lean manufacturing initiative. 70% reduction in shipping errors. All shipping people are out of office by 5pm every day. Dock-to-dock cycle time metric improved from 1.5 weeks to 4 hours. Sufficient improvement in productivity to enter new cost-sensitive markets that previously were closed to this company.
Track material through all process steps. Regulatory compliance, plus detect product loss, measure asset utilization, track amounts and costs of materials, utilities, track waste discharge, yields. Secure, accurate data for accounting.	Improved profitability while meeting government regulations. Reduced material loss from 2.5% to 0.75%. Reduced inventory. Caught problem before it left facility. Avoided capital expenses by improved equipment utilization.

Interesting Initiatives for Discrete Manufacturing Professionals

Trends such as Lean manufacturing arise because one company learns to do something better than its competitors; and the word gets out. Other trends occur as a result of social and technological developments outside of the world of manufacturing companies. Globalization is an example of this. Improvements in information technology with rapid development of economies and infrastructure in countries such as China, India, and Brazil have created unprecedented opportunities for companies to source materials and services from anywhere in the world, manufacture anywhere, and sell anywhere.

Trends also arise when leading manufacturers meet with government agencies and universities to work on particular problems of joint interest. Examples of these initiatives for discrete manufacturing as a whole are:

1. Workshop for Smart Assembly in October 2006. The United States National Institute of Standards and Technology (NIST) conducted the workshop with representatives from Ford, GE, Boeing, GM and others. Contact Dale Hall, Director, Manufacturing Engineering Laboratory, NIST.

2. Federal Interagency Working Group on Manufacturing R&D identified "Intelligent and Integrated Manufacturing Systems" as one of three critical areas of national need.

3. United States National Science Foundation (NSF) Industry/University Cooperative Research Center on Intelligent Manufacturing Systems. Ford and Toyota participate. The goal is zero breakdown. Contact Jay Lee, Ohio Eminent Scholar and L. W. Alter Chair professor in Advanced Manufacturing, University of Cincinnati, Jay.Lee@uc.edu, U. of Michigan and U. of Missouri-Rolla are also involved.

4. The Automotive Industry Action Group (http://www.aiag.org – offices in Southfield, Michigan, USA and Shanghai, China) also has teams of people from automotive companies and often government agencies and other consortia working together on joint initiatives:

   » Inventory Visibility & Interoperability (IV&I):
   - Phase 1 – MIN/MAX and Basic Web Services Profile
   - Phase 2 – eKanban and Reliable, Secure Messaging
   - Phase 3 – Vendor Managed Inventory (VMI) and Reliable, Secure Messaging

   » Early Warning Standards – Warranty
   » Plant Floor to Business (P2B)
   » Material Off-Shore Sourcing

Drivers for Flexibility, Agility and Responsiveness

Standards for communication between plant systems and the enterprise or supply chain evolved over the last 15 years, from high-level business process models to data exchange schema to transaction sequences to defined message structures.

However, industry has been slow to apply them due to:

- Lack of vertical industry instance or templates of the standards
- Cost of replacing their established legacy system based on disparate data models and point-to-point interfaces.

Some progressive companies have realized large benefits by establishing a standards-based baseline for application and integration utilizing a common canonical schema and transaction set. Like most CI initiatives, this change in IT architecture requires a 3–5 year migration of applications and interfaces to the baseline. This CI migration is significantly accelerated through SOA technologies. This reduces the high cost in both initial investments and maintenance of diverse systems.

In 21st Century manufacturing, bidirectional information exchange must occur between business and the shop floor. Data aggregation rules and analytics are now being developed to provide access to only the data needed and authorized. The aggregated information is delivered through role-based dashboards, portals, and handheld-devices customized and consistent to each person’s need, action and security level.

Globally distributed supply chains require open, yet secure communications with suppliers wherever they are in the world. With many plant operations based on paper-based transactions and isolated plant and office applications, one of a company’s challenges to globalization is normalizing and integrating fragmented and overly-complex IT architectures with different infrastructures from region to region, company to company, plant to plant, and line to line. To achieve the required adaptability, this plethora of disparate enterprise and supplier systems are being simplified through a transformation to standards-base SOA.

The Automotive Industry Action Group’s (AIAG) Plant Floor to Business (P2B) project’s business case summarizes the need as follows.

> “There are no broadly accepted and implemented standards for communication between plant systems and the enterprise or supply chain. This is causing high cost in both initial investments and maintenance of diverse systems. Plant interoperability is becoming even more critical especially with:
>
> - Globalization demands real-time communication from the plant floor seamlessly throughout the distributed supply chain network.
>
> - More complex manufacturing systems and a more complex product mix
>
> - Suppliers located both in supplier parks and in their customer’s plants call for more data integrity and security from their competitors.
>
> - The need to exchange data between the boardroom and shop floor, but only get access to data needed or authorized. Dashboards are all customized and consistent to each person’s need and role in business processes.

- Open, yet secure communications with suppliers wherever they are in the world since suppliers are a critical part of the ecosystem. Supplier parks, supplier integration into the OEM manufacturing facilities and OEMs deploying their systems in supplier plants are all creating expense for both the OEM and the supplier.
- Fragmented, over complex IT architectures and infrastructures from region to region, company to company, plant to plant, and line to line and the plethora of enterprise and supplier systems.
- Need to improve cost, time and quality in plant and enterprise systems.
- Recalls and warranty costs must be brought under control, through emphasis on quality of product, process, parts traceability, and improved inventory management throughout the supply chain.
- Sequencing material to minimize inventory buffers which is a key to Lean and associated cost savings. This means that information flow and integration is critical to have the right parts in the right place at the right time.
- The cost of system maintenance and repairs with many divergent systems is too high.
- There is no standard means of connecting and communicating across all the various PLCs, despite the important PLC interfaces in every plant.
- RFID and wireless devices in the plant add to the complexity of integration.
- Need to rapidly integrate with existing systems as new systems are brought on-line.

P2B interoperability is key to synchronization and visibility. Industry standards cut costs and time. Senior executives from the leading software companies, including Microsoft, Oracle, IBM and SAP are very supportive of standards. The standards need further work, and methodologies for interoperation need to be developed.

Convergence of plant to business standards is slow but significant. End user voices and participation are essential for effective analysis, decisions and planning regarding standards activities. Smaller material suppliers' capabilities and support for emerging standards is constraining the rate of adoption of those standards.

SYSTEM VIEW OF A MANUFACTURING COMPANY
Environment

Manufacturing IT Infrastructure Landscape

In physical terms, there are three areas in the plant where information technology is deployed:

- Office area
- Shop floor
- Local plant data center (plant computer room or campus-wide data center)

This is a typical scenario. People performing office activities in plants make use of IT solutions and infrastructure that are common at all sites in the enterprise. Office workers use standard client PCs attached to the Office Automation (OA) network. These are primarily located in the office area of the plant and in workgroup meeting rooms near the shop floor. The OA network is available on the shop floor as well.

Programmable equipment, Programmable logic controllers (PLCs), coordinate measuring machines (CMMs), material handling equipment, test stands, and other devices exist throughout the shop floor, while mobile devices may be used in the yard around the plant. These devices connect to special purpose industrial networks which are segmented from the OA network for personnel safety and data security.

Each plant has a computer room operated as a small Data Center with a raised floor, multiple independent power sources, uninterruptible power supply (UPS), air conditioning, etc. Servers and other devices in the computer room are available to clients on the OA network and may be accessible from the specialized industrial networks on the shop floor. In some instances, the servers are purposely not accessible to clients on the OA network.

Wireless Networks and Collaboration

Shop floors utilize a highly diverse set of technologies: Radio frequency identification (RFID), tags, sensors, etc. Both wireless and copper/fiber internet protocol (IP) networks provide transport services for collaboration using data, voice and multi-media both on the shop floor and in the office. Voice over Internet Protocol (VoIP) for telephone communications, web cameras for video-conferencing, and instant messaging using mobile devices are being introduced, but security and safety are concerns. Infrastructure is still developing to secure collaboration methods between people at all locations in the plant and surrounding yard, staff engineers, centrally located product design teams, and material suppliers.

Clients and Mobile Devices

Standard office automation technology should be used wherever possible. However, plants may also need office printers, a variety of specialized plotters, label and barcode printers. There may be diverse client form factors for use on the shop floor. Many times, industrial built, rugged devices are required; however, they usually run a commonly available operating system (OS) (Windows CE operating system, Embedded Java, etc).

This domain typically includes:

- Thin client hardware for kiosks/walkup stations
- Clients for equipment controls (no human user interface)
- Clients with human user interfaces that control machinery (e.g., test stands)
- Machine tools that embed PC-based controls

Plant Floor Device Integration

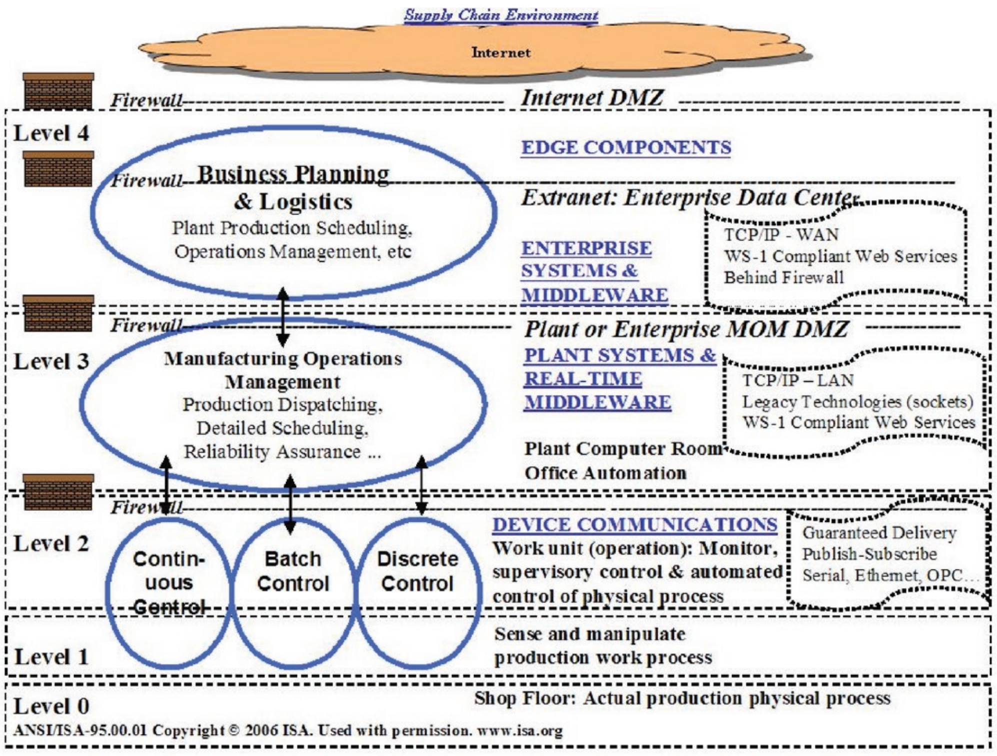
For existing equipment and process controls, an OPC®[1] (OLE for Process Control) server may be used to convert data communicated from devices using proprietary protocols into standard messages for communication with transactional applications. Over time, XML Web Services as defined by the Web Services-Interoperability organization (WS-I), using the latest security and reliable messaging specifications, may be used to integrate with controls. The OPC Foundation has defined a new specification along these lines called "OPC Unified Architecture" (OPC UA). OPC UA has specified several implementations including J2EE, Microsoft .Net, and OPC UA Binary. Using Web Services and OPC servers enables the infrastructure as a whole to be refreshed, common and standard while supporting a wide variety of shop floor technologies which often have lives of 10 to 15 years or more.

Figure 1 shows a logical view of a Supply Chain Enterprise. The enterprise is separated into various layers (corresponding to the ISA-95 Enterprise Domain Hierarchy) of the business: Shop Floor, Plant Computer Room (Office Automation), Plant or Enterprise Manufacturing Operations Management DMZ, Enterprise Data Center, Extranet, Internet DMZ and Internet. The logical separation between the layers is either a communication bus or a firewall with a communication bus connecting all layers and a firewall between the appropriate layers. The various communication buses may be one physical implementation or several implementations. The separation shown in the figure is used to specify the type of communication used at a particular layer. For instance, at the device communication layer, OPC and other types of architecture (i.e. sockets) exist for communicating between devices such as PLCs.

Level 4 controls and monitors inventory levels and also establishes the basic plant schedule—production, material use, delivery, and shipping. System time frames are in months, weeks, and days.

Level 3 establishes work unit definition and control, in the form of workflow/recipe control, to produce desired end products. Manufacturing Operations Management (MOM) systems including Manufacturing Execution Systems (MES) analyze work data, maintain records and optimize the production process. System Time Frame is in days, shifts, hours, minutes, and seconds.

Figure 1: ISA-95 Domain Hierarchy from Purdue Reference Model


Supply Chain Environment
Internet
Firewall
Internet DMZ
Level 4
EDGE COMPONENTS
Firewall
Business Planning & Logistics
Plant Production Scheduling, Operations Management, etc
Extranet: Enterprise Data Center
ENTERPRISE SYSTEMS & MIDDLEWARE
TCP/IP - WAN
WS-1 Compliant Web Services
Behind Firewall
Firewall
Plant or Enterprise MOM DMZ
Level 3
Manufacturing Operations Management
Production Dispatching, Detailed Scheduling, Reliability Assurance ...
PLANT SYSTEMS & REAL-TIME MIDDLEWARE
TCP/IP – LAN
Legacy Technologies (sockets)
WS-1 Compliant Web Services
Plant Computer Room
Office Automation
Firewall
Level 2
DEVICE COMMUNICATIONS
Work unit (operation): Monitor, supervisory control & automated control of physical process
Guaranteed Delivery
Publish-Subscribe
Serial, Ethernet, OPC…
Contin-uous Control
Batch Control
Discrete Control
Level 1
Sense and manipulate production work process
Level 0
Shop Floor: Actual production physical process
ANSI/ISA-95.00.01 Copyright © 2006 ISA. Used with permission. www.isa.org

Level 2 defines the work unit (operation). Systems monitor and provide supervisory control and automated control of the production work process. System Time Frame is in hours, minutes, seconds and micro seconds or less.

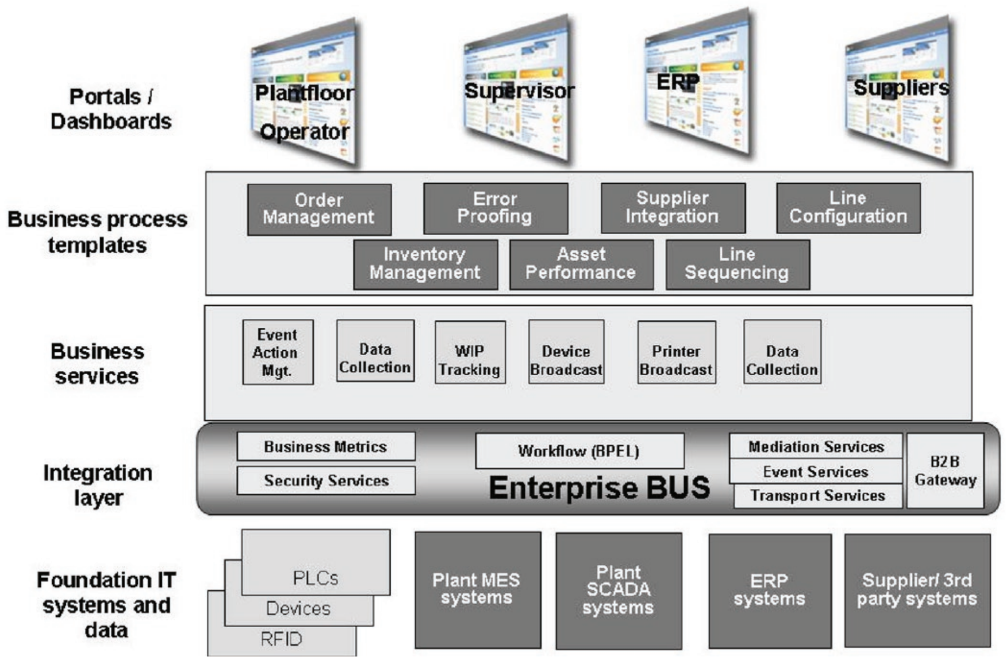
Level 1 defines the sensing and manipulating of production work process. Device Time Frame is in real-time; in micro seconds or less.

Level 0 defines the shop floor through the actual production physical process.

Many companies have a policy that the plant must keep running 24 by 7, even when WAN and Internet connections are unavailable. In this environment, production-critical systems must be hosted in the plant. This policy is most likely to apply to capital-intensive, high volume production facilities where the supply chain functions as a continuous process model.

Layered Enterprise View

Figure 2: Overall SOA for Manufacturing Vision



A services oriented architecture is shown as functional layers, as shown in Figure 2, Overall SOA for Manufacturing Vision. The bottom layer consists of the existing or legacy applications that provide the foundation for how the business' data is used. These are the 'mission critical' applications that keep the business running on a day to day basis. The next layer provides the integration. In an SOA, the integration layer is realized with an enterprise services bus (ESB). The ESB layer provides security, transport, mediation and event services. It also can provide business metrics and a workflow or business process engine. The Business Services (middle) layer is an abstraction layer of services that 'front' the foundation IT systems. These services are what do the work within the SOA. These services are represented using Web Service Description Language (WSDL) that wraps the business applications. The Business Processes layer consists of business processes that are created by combining the services in the Business Services layer together to create composite applications. Composite applications are a new way to do application development within the SOA (discussed in more detail in the Process Choreography area of the SOA Overview section). The Portal/Dashboard layer consists of data aggregation and visualization.

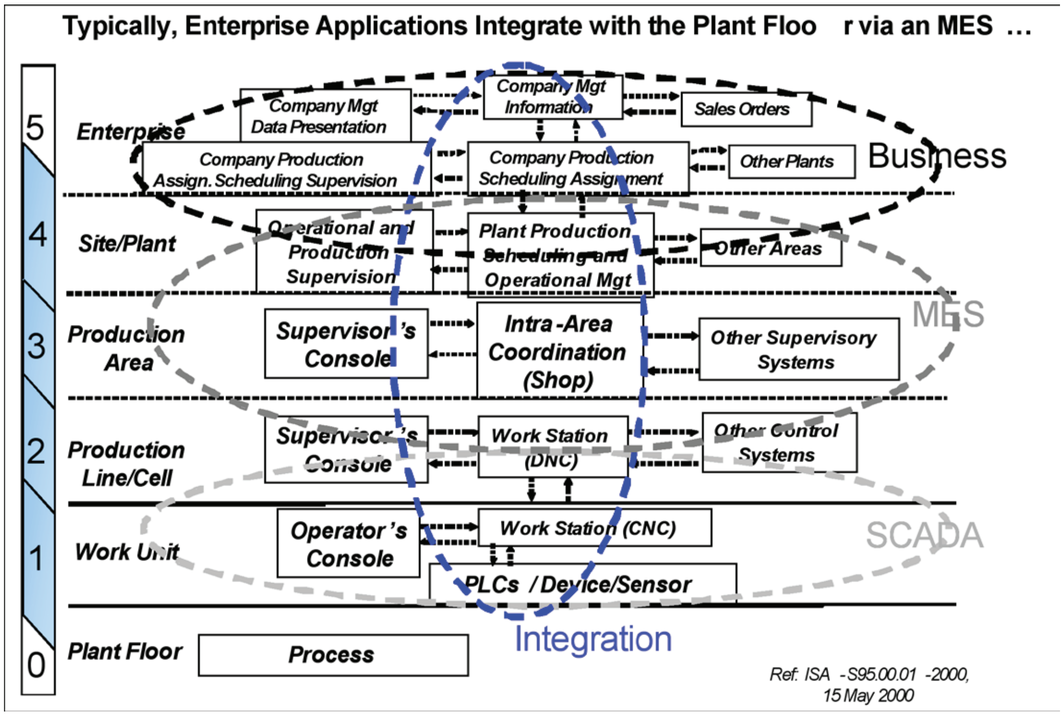
When using services within an SOA, those services can be called to:

1. Perform business tasks via process choreography or directly
2. Be used within a portal or web application for data aggregation or visualization
3. Create new visual applications which trigger services or multiple services (via a composite application)
4. In a similar way, the enterprise as a whole can be represented as layers under the control of three basic types of applications:

5. Business or Enterprise Applications such as those often found in ERP for financial reconciliation and reporting, order management, materials masters, etc
6. MES or MOM applications which coordinate production and integrate with the plant floor and with enterprise level applications
7. Device layer applications such as distributed control system (DCS) or supervisory control and data acquisition (SCADA) which are used to manage the device layer in manufacturing

Figure 3, Typical System Structure of Manufacturing Enterprise, is based on the Enterprise Domain Model from ISA-95, Enterprise-Control Integration Standard (see Figure 1) and shows the general orientation of these three classes of applications.

Figure 3: Typical System Structure of a Manufacturing Enterprise


Typically, Enterprise Applications Integrate with the Plant Floo r via an MES …
5
Enterprise
Company Mgt Data Presentation
Company Mgt Information
Sales Orders
Company Production Assign. Scheduling Supervision
Company Production Scheduling Assignment
Other Plants
Business
4
Site/Plant
Operational and Production Supervision
Plant Production Scheduling and Operational Mgt
Other Areas
3
Production Area
Supervisor's Console
Intra-Area Coordination (Shop)
Other Supervisory Systems
MES
2
Production Line/Cell
Supervisor's Console
Work Station (DNC)
Other Control Systems
1
Work Unit
Operator's Console
Work Station (CNC)
PLCs / Device/Sensor
SCADA
0
Plant Floor
Process
Integration
Ref. ISA -S95.00.01 -2000, 15 May 2000

Today, most manufacturing companies can be described using the ISA-95 Enterprise Domain Model. However, companies are now re-thinking how data is integrated, or how data flows, between these levels. SOA can play a large part in how this integration takes place.

Manufacturing 2.0

AMR Research has developed an SOA for manufacturing approach that explains the manufacturing specific requirements. There are differences between the SOA utilized for enterprise through an ESB and the SOA utilized by the near real-time manufacturing operations management systems in a plant through a Manufacturing Services Bus (MSB). The manufacturing operations specific requirements for SOA are called Manufacturing 2.0 which differentiates from the so called Manufacturing 1.0 architectures based on standalone client/server data base applications that attempted to represent business process modeling through point-to-point interfaces and custom data transformation between applications. This section provides a brief introduction to Manufacturing 2.0 and other SOA elements. The SOA elements and mechanisms will be addressed in the later sections of the paper in the more details. The Manufacturing 2.0 explanation is simply intended to provide the reader with a holistic overview of how SOA may be applied at the enterprise and production facilities.

Figure 4 is AMR's representation of Figure 2, Overall SOA for Manufacturing Vision, where the "Mfg Ops" (manufacturing operations) represent the Plant MES and Plant SCADA from Figure 2. Figure 4 is the base reference for the Manufacturing 2.0 explanation in Figures 5-7. The "Mfg Ops" element in Figure 4 is than exploded in Figure 5, Manufacturing SOA: Foundation for Manufacturing 2.0, to provide the detailed elements and relationships of Manufacturing 2.0 SOA that enables manufacturing operations within and across production facilities.

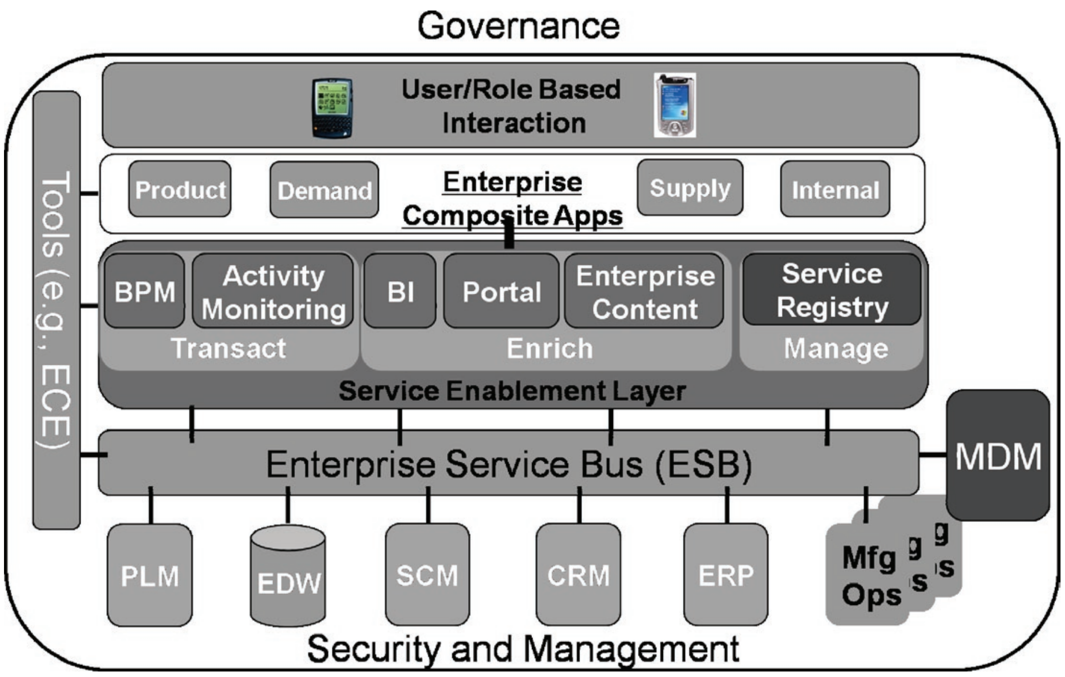
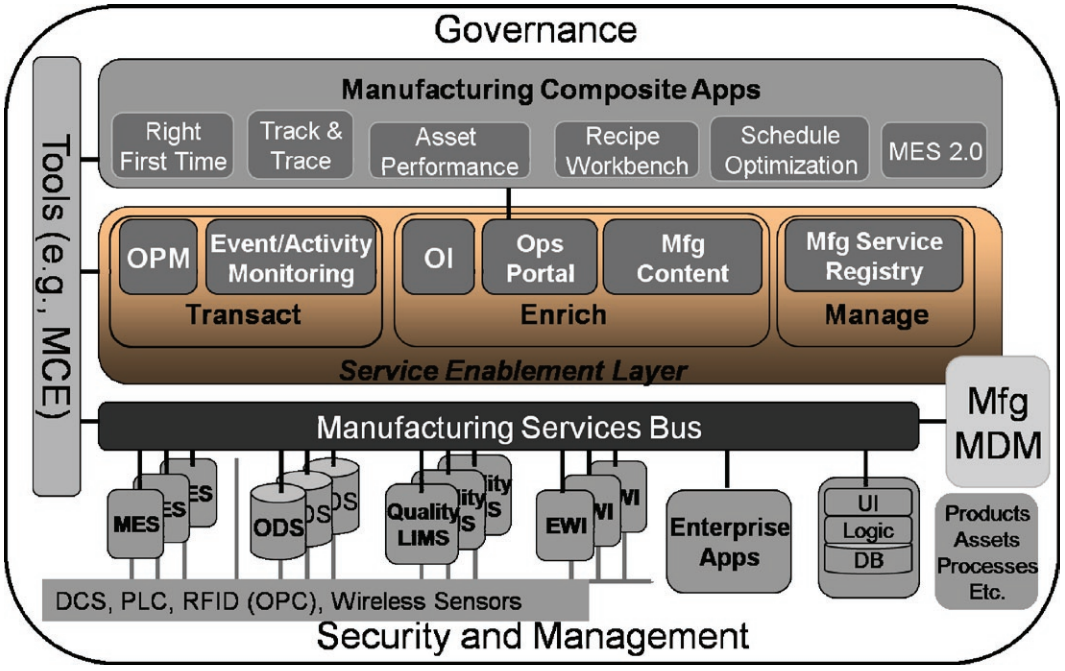


Figure 4: Enterprise SOA: A Work in Progress

Copyright © 2008 AMR Research: All rights reserved. ECE = Enterprise Composition Environment, BPM = Business Process Management MDM = Master Data Management, EDW = Enterprise Data Warehouse

Figure 5: Manufacturing SOA: Foundation for Manufacturing 2.0

Copyright © 2008 AMR Research: All rights reserved. MCE = Manufacturing Composition Environment, OPM = Operations Process Management, OI = Operations Intelligence, LIMS = Laboratory Information Management System, EWI = Electronic Work Instructions

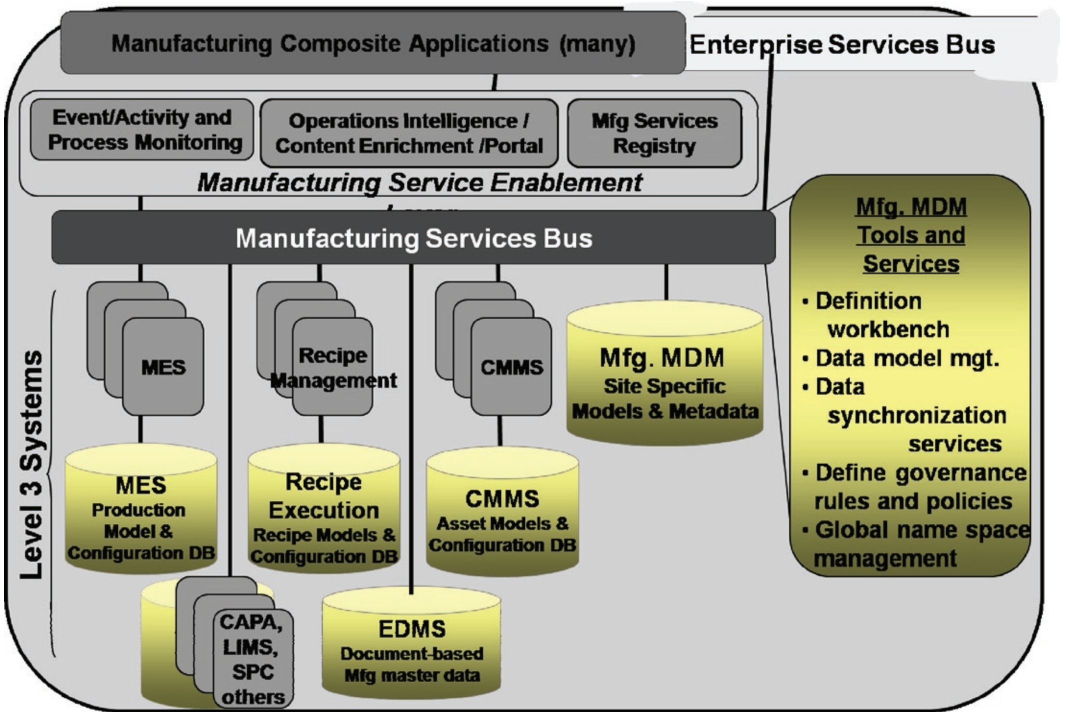


The paper simply introduces the Manufacturing 2.0 SOA approach as one possible evolutionary path for the application SOA in support of the near real-time operations applications of a plant. Please refer to the original works from AMR for more details than this high level breakdown provides. Figure 5 highlights that separate manufacturing services bus (MSB) is required due to high transactions, high parametric data load and near real-time requirements for operations applications. The MSB may be scaled down to a plant or area of plant or across multiple production facilities depending on the transaction/data load and response requirements of the operations workflows being supported by the plant applications. A key aspect of Mfg. 2.0 is the explanation that the manufacturing master data management (Mfg MDM) is different than MDM on the ESB for the enterprise business processes. The Mfg MDM services a different set of applications for manufacturing operations management such as dispatching, route execution, and alarm & event applications which have a much more granular set of objects, attributes, and production rules than the MDM that is representing the enterprise planning, (master) scheduling, and logistics. However, MDM is too large and complex of an architectural topic to be addressed adequately in this paper. The MDM topic is only addressed in this section with a basic definition as an identified critical SOA design component. For a detailed explanation, a highly recommended MDM reference is "Enterprise Master Data Management: An SOA Approach to Managing Core Information", http://safari.oreilly.com/9780137149674

The form and role of MDM is very dependent on the vertical industry, products set, market segment, production type and complexity, and supply chain type. For instance, MDM is much different from life sciences to automotive to aerospace to electronics. This is illustrated in Figure 6, Manufacturing MDM in Manufacturing

SOA Architectures, in the site specific model and metadata. Due to the high change rate of the Mfg 2.0 applications due to new product introductions, change SKU counts, evolving process technologies documents, and production scaling, Mfg MDM requires following dedicated set of tools and services:

- Definition workbench
- Data model mgt.
- Data synchronization services
- Define governance rules and policies
- Global name space management



Manufacturing Composite Applications (many)
Enterprise Services Bus
Event/Activity and Process Monitoring
Operations Intelligence / Content Enrichment /Portal
Mfg Services Registry
Manufacturing Service Enablement
Manufacturing Services Bus
Mfg. MDM Tools and Services
• Definition workbench
• Data model mgt.
• Data synchronization services
• Define governance rules and policies
• Global name space management
Level 3 Systems
MES
Recipe Management
CMMS
Mfg. MDM Site Specific Models & Metadata
MES Production Model & Configuration DB
Recipe Execution Recipe Models & Configuration DB
CMMS Asset Models & Configuration DB
CAPA, LIMS, SPC others
EDMS Document-based Mfg master data

Figure 6: Manufacturing MDM in Manufacturing SOA Architectures

Copyright © 2008 AMR Research: All rights reserved. mMDM = Manufacturing MDM, CAPA = Corrective Action Preventative Action, SPC = Statistical Process Control

Figure 7, Manufacturing SOA as a part of Enterprise SOA, clarifies the relation of the Manufacturing SOA role in the Enterprise SOA. The Mfg. 2.0 is a SOA approach, not an application that embraces diversity, complexity, and new dynamism in demand-driven manufacturing, while:

- Leveraging scarce manufacturing talent
- Leaving and leveraging manufacturing investments rather than ripping and replacing with better “mouse traps”
- Removing software usability and rigid data models as barriers to construction, deployment, and adoption of manufacturing applications
- Intra- and inter-enterprise collaboration on products and manufacturing processes

Applications can be adapted to changing business processes, easily and inexpensively in this approach. Emerging Mfg 2.0 characteristics are:

- User-centric interfaces that:
  - Streamline activities
  - One-click navigation and drill-down access to functionality from related applications through a single cohesive interface (i.e. quality management and MES, Overall Equipment Effectiveness (OEE) and asset management, corrective action preventative action (CAPA) and laboratory information management system (LIMS), LIMS and MES
  - Take advantage of available shop-floor talent for deployment, reconfiguration, and software maintenance
  - The Microsoft usability model, guided procedures
  - Provide familiar interaction paradigms and relevant information models
  - Graphical asset models, in-plant GPS, navigation, and mapping capabilities, use of multimedia and tagging

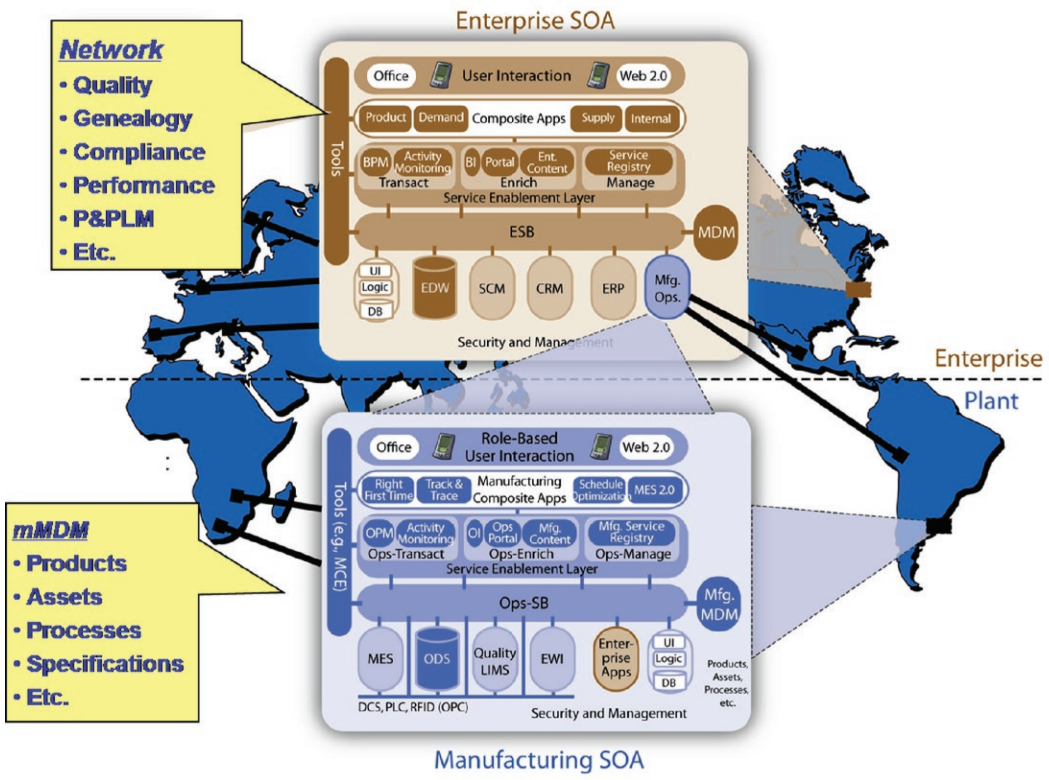
Applications work the way users "think" about their work through mapping application functionality to industrial business process or workflow management.

Other emerging Mfg 2.0 characteristics are:

- Manufacturing architectures that:
  - Capitalize on existing investments by using manufacturing SOA instead of ripping and replacing them with monolithic applications
  - Leverage ISA S95/OAGIS models for manufacturing services
  - Utilize manufacturing services bus architectures
- Incorporation of Web and Enterprise technology constructs such as blogs, instant messaging, mash-ups, search, tagging, wikis, and cool, always-connected mobile devices
- Engage younger, Web-savvy generations in manufacturing, enhance software usability, collaboration, and knowledge sharing
- Support for low-cost tags, intelligent sensors, pervasive networks, and mobile workers
  - Mobile users interact directly with the production environment through mobile asset management, re-mixed quality, MES, and other new composites deployed on handheld devices

Figure 7: Manufacturing SOA as a part of Enterprise SOA

Copyright © 2008 AMR Research: All rights reserved. P&PLM = Product Process Lifecycle Management



- Event-driven, supply network collaboration (intra- and inter-enterprise) platforms
  - The Web as a platform offers unprecedented opportunities to dissolve boundaries between suppliers and manufacturers, manufacturers and their contract partners
- Convergence of product data management and process development models
  - For rapid development of new products and manufacturing processes to accelerate time to market

SERVICE ORIENTED ARCHITECTURE – AN OVERVIEW
What is SOA?

SOA Evolution

SOA is the culmination of various integration strategies over the years. As the IT industry moved away from, or found alternatives to, the mainframe computing paradigm, smaller more localized computers began to emerge. As this infrastructure of smaller, more localized, computers began to grow, a need to make these machines communicate with each other evolved. This need to have co-located computers "talk" with each other, first, brought about the client/server architecture and then distributed systems architecture with distributed data and functions from roughly 1995 to 2005.

Point-to-Point Integration

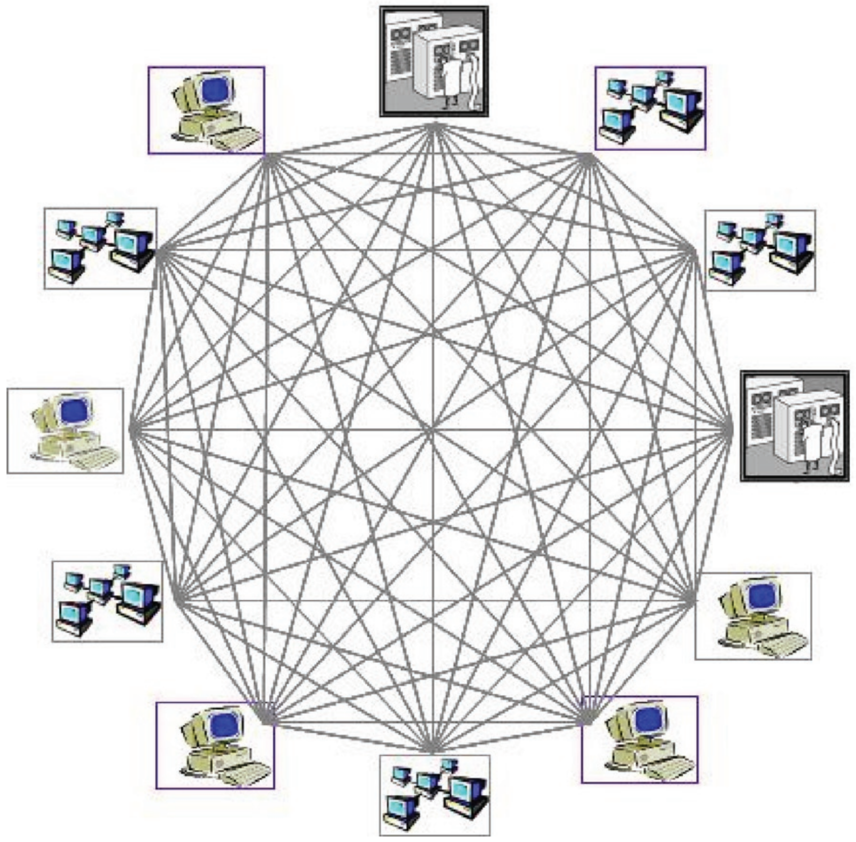
Point-to-point integration between a client and server was, in the early days of system integration, the main method of communicating. The client had to know about the server, and the server had to know about the client. In addition, each of the integrated computers had to use the same protocol and then "language translation" in order to communicate.

The client/server and its point-to-point strategy brought about what today is called the 'spaghetti integration' picture. Figure 8 below shows how complex a company's application integration can get with a point-to-point messaging strategy. It is necessary to define up to sixty-six different interfaces in order to integrate the twelve applications pictured. Each client/server system has its own language schema or data model. If another application is added, every application needs to communicate with that new application through an additional custom interface. Each interface does a different language translation. One can see how the maintenance and support of this architecture can become unwieldy very quickly. Especially in manufacturing, Level 3 Message Oriented Middleware applications and their numerous interfaces are in constant, dynamic change due to new product introductions (NPIs), scaling up from pilot to full volume, scaling down of old products, and continuous improvement or advancement of manufacturing processes.

Enterprise Application Integration (EAI)

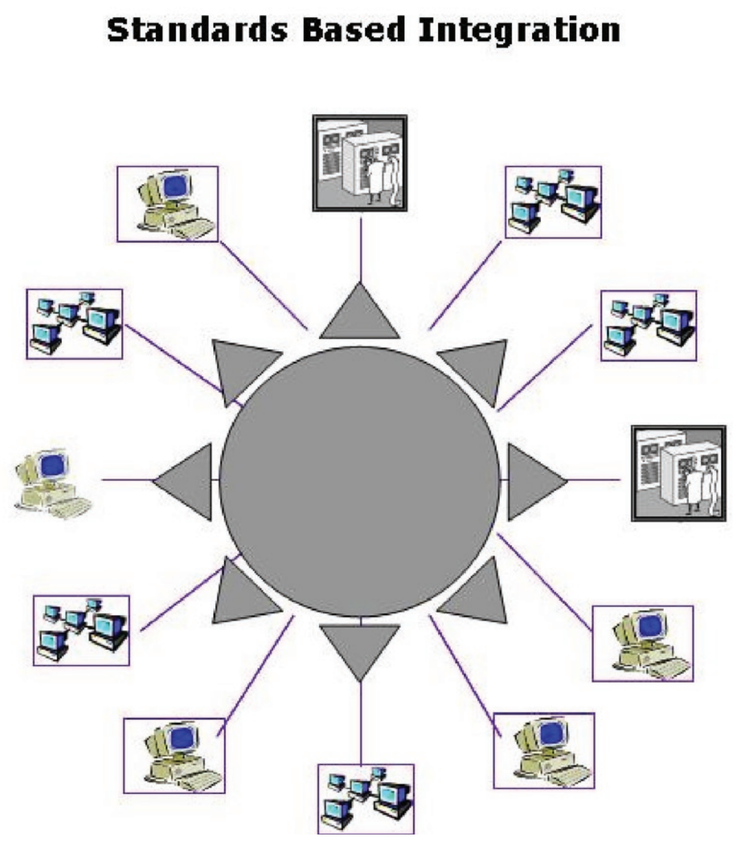
In the late1990's, in an effort to simplify point-to-point communication, a new strategy called Enterprise Application Integration (EAI) became popular. This is the so called "hub-and-spoke" method of integration, where all communication comes through one central point before arriving at its final destination. The hub is responsible for routing, language translation, transaction sequencing, and mediating content and protocol of the message before ultimately sending it on to its final destination. This EAI approach was much simpler to create and maintain than the point-to-point approach. With the hub and spoke approach, the sender only needs to know how to communicate with the hub rather than having to know how to communicate with n-number of endpoints. In essence with this approach, the logic for message routing and content and protocol mediation is moved from the sending and/or receiving machine to the hub. Therefore, the applications (spokes) need only be concerned with the business logic they are to provide rather than how to communicate to multiple endpoints. As data is passed through the hub, it is generally converted from an application-specific format to a common canonical format. As it passes out of the hub, it is generally converted to the application-specific format of the receiving application.

Figure 8: Point-to-Point Integration



As seen in the Figure 9, Hub-and-Spoke, the sixty-six interfaces from Figure 8 are reduced to just twelve. The same twelve applications only need to know the interface for the messaging hub. The hub is responsible for knowing how to communicate with any of the endpoint applications.

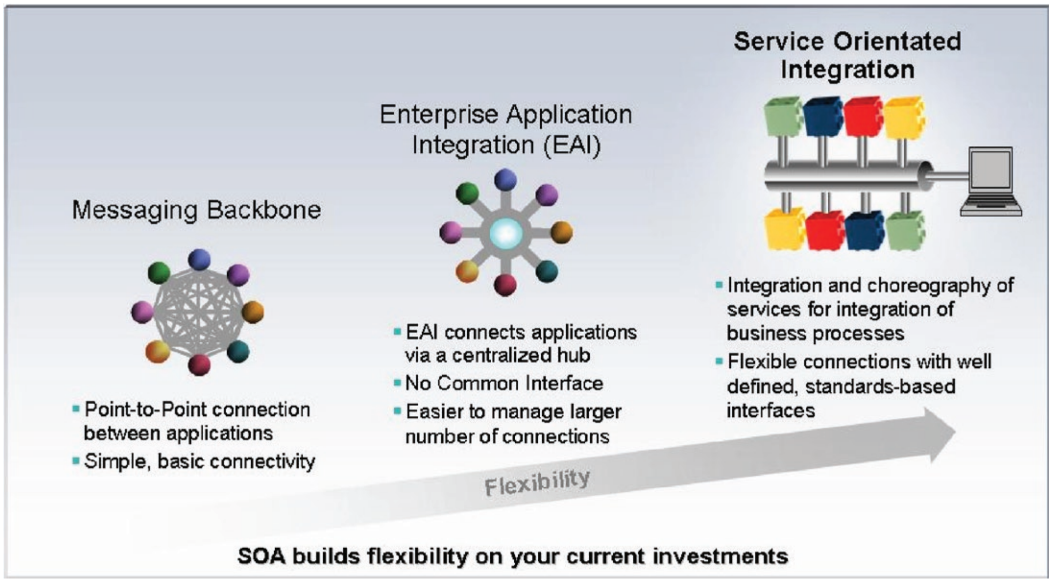
Figure 9: Hub-and-Spoke Integration



With the hub-and-spoke method, the routing, mediation and transformation logic in the hub can also get difficult to maintain without complex configuration management practices. There is a tendency for the business logic to creep into the hub which makes changing that logic more difficult as more and more content builds up in the hub. Also, with EAI, you are still dealing with large, monolithic, and disparate applications that communicate in different ways.

While the EAI approach is an improvement over the point-to-point messaging backbone approach, EAI still leaves gaps that a standards-based SOA fills. Figure 10, Evolution of Integration, illustrates how integration has evolved over time. The industry is now moving toward a services-based approach to integration, where smaller, self describing pieces of logic interact and communicate together on an Enterprise Services Bus (ESB). The ESB is one of the major components of a services oriented architecture.

Figure 10: Evolution of Integration

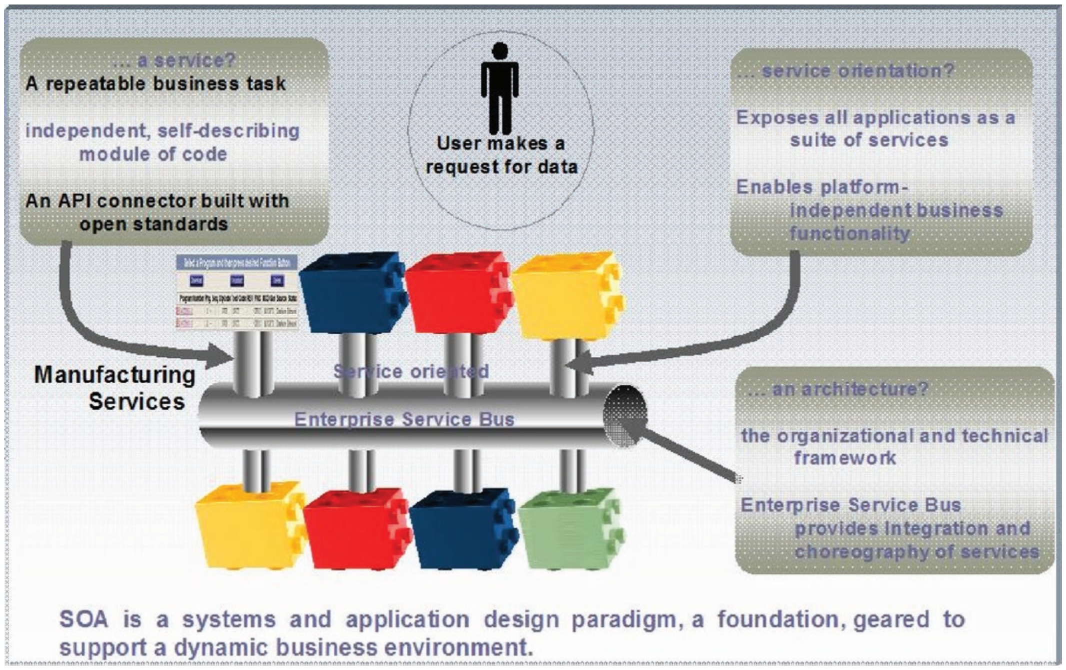

Messaging Backbone
Point-to-Point connection between applications
Simple, basic connectivity
Enterprise Application Integration (EAI)
EAI connects applications via a centralized hub
No Common Interface
Easier to manage larger number of connections
Service Orientated Integration
Integration and choreography of services for integration of business processes
Flexible connections with well defined, standards-based interfaces
Flexibility
SOA builds flexibility on your current investments

Services Oriented Architecture (SOA)

SOA is an IT systems model which gives companies flexibility in the way they create business applications. SOA is not just focused on application integration, but also application construction from existing IT assets. The architecture allows for the creation of composite business applications from independent, self-describing, and interchangeable code modules called services. These services are available for use on a services bus and can be arranged together, into a business process, or composite application, using process choreography. So, the major components of SOA are:

- Services
- Services Bus
- Process choreography – composite applications
- Message transformation, mediation and routing
- Services Registry

Figure 11: SOA Overview


… a service?
A repeatable business task
independent, self-describing module of code
An API connector built with open standards
User makes a request for data
… service orientation?
Exposes all applications as a suite of services
Enables platform-independent business functionality
Manufacturing Services
Service oriented
Enterprise Service Bus
… an architecture?
the organizational and technical framework
Enterprise Service Bus provides Integration and choreography of services
SOA is a systems and application design paradigm, a foundation, geared to support a dynamic business environment.

Components of a Services Oriented Architecture

Services:

Services are the core building block within an SOA. Defining what a service actually is can be difficult. Some definitions of a service which have become prevalent are:

- A repeatable business task
- An independent, self-describing, module of code
- A discoverable resource that executes a repeatable task, and is described by an externalized service specification.
- In lieu of coming up with a bullet proof definition of what a service is, let us discuss some key concepts of services within an SOA.

Services are not based on IT concepts but rather they are based on business concepts.

With SOA, the idea is that business drives the IT systems, rather than the IT systems dictating how the business runs. This brings about flexibility, one of the main benefits often touted for using SOA.

Services are self described in a standard way using web services definition language (WSDL). The WSDL provides a standard semantic for specifying a service's interface, operations, policies, etc…

Services are reusable.

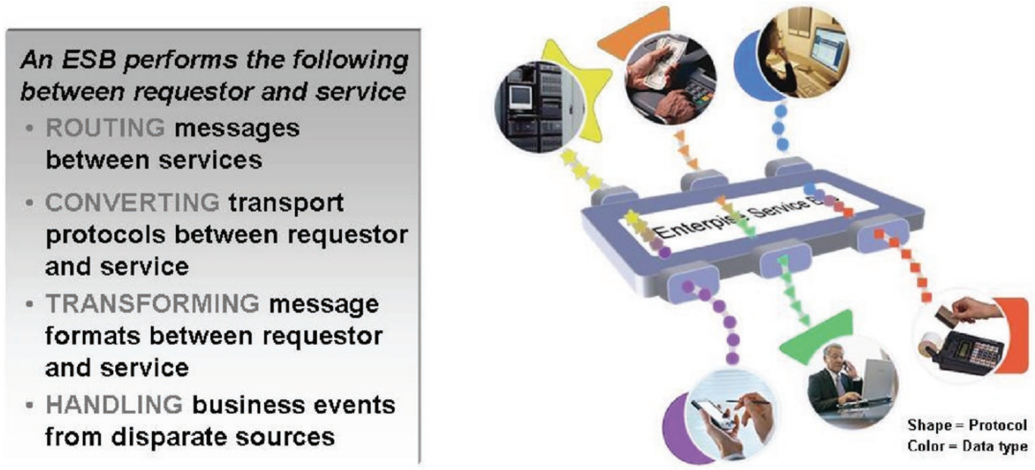
The "art" of defining/creating services is in determining the granularity of the services created. The degree of granularity determines the amount of reuse attained by the service.

Services are self contained and independent.

Services can either "stand" on their own, without any external dependencies or be combined with other services to create a composite application. Both examples are outlined throughout this paper.

Service Bus:

Figure 12: What is an ESB?



An ESB performs the following between requestor and service

- ROUTING messages between services
- CONVERTING transport protocols between requestor and service
- TRANSFORMING message formats between requestor and service
- HANDLING business events from disparate sources

An Enterprise Service Bus (ESB) is a flexible connectivity infrastructure for integrating applications and services. One of the objectives of an ESB is to reduce the number, size, and complexity of interfaces in a SOA. An ESB is not a software product. It is, however, a new way of looking at how to integrate applications, coordinate resources and manipulate information. The ESB enables software written in different programming languages, and running on different platforms, to connect.

How Does ESB Solve the Point-to-Point Integration Problem?

The ESB provides a central point for connecting distributed applications. This is unlike many previous approaches such as remote procedure call (RPC) or distributed objects. The ESB uses a pattern to enable connection between software running in parallel on different platforms, written in different programming languages and using different programming models.

Process Choreography:

When SOA is employed as an integration strategy, it brings about a catalog of self-describing, atomic business services that are used together to create a business process. Often, a business event (e.g. a vehicle arriving at an assembly line work station) will need to trigger interactions between many different manufacturing and/or business systems. These interactions can be achieved using process choreography within an SOA.

Process choreography allows for multiple business services to be used together, in combination, to implement a business process. Within an SOA, process choreography is realized via BPEL (Business Process Execution Language). In general, BPEL is a standard XML-based language which, when compiled into executable code, allows one to compose business processes out of available services. BPEL is described in more detail in the next section.

An example of a business process triggered in a manufacturing area would be a WIP (work in process) tracking process. In an automotive example, when a vehicle enters an assembly area work station, an event triggers a business process that runs the following services:

- Update vehicle location in the MES application
- Update vehicle history record in the MES application
- Obtain vehicle options from the vehicle order information
- Broadcast vehicle options to future assembly work station PLCs
- Broadcast build instructions to a local printer

The above process can also be called a composite application. Many different composite applications can be created using the available services in the catalog. These composite applications can also be represented as single, more complex, services and then used within other business processes. SOA enables this new way of application development (or application assembly). It is no longer necessary to build large, monolithic applications, with stringent APIs. SOA allows for applications to be built dynamically by either assembling services sitting on top of already existing or legacy applications, or by assembling services created from newer, services-based applications.

The services execute using a standards-based BPEL process running within a process engine. IBM, BEA, Oracle, GE Fanuc, SAP and other companies offer BPEL execution engines with their application server products. These services are also available to be consumed in other business processes.

Message transformation, mediation and routing:

- The ESB in an SOA is responsible for the classical EAI tasks inherent in integration applications, primarily:
- Protocol switching (e.g. JMS in, SOAP out and vice versa)
- Message transformation (e.g. fixed record structure in, OAG BOD out)
- Message mediation (e.g. conversion of values understood by SAP to values understood by a plant floor system)
- Message routing (e.g. message received from plant floor needs to be transformed and sent to three other systems)
- Security, logging, transactional support

Service Registry:

As an organization moves along the path to SOA, an increasing number of services will be identified. Over time, it will become important to have a central location to register and manage the services, as shown in Figure 13. A service registry can serve as the central repository for service endpoint descriptions (in WSDL) as well as for metadata about the services. The registry can provide version control and change management. This can be viewed much as a library control system for SOA development.

Universal Description, Discovery, and Integration (UDDI) is one example of a registry specification that defines the ability to store and retrieve information about web services. Service Registry products are available from many vendors, including Sun, Microsoft, and IBM.

A service registry can be used to find, publish, manage and subscribe to services. A registry may also be used to store metadata, such as:

- Service provider
- Availability
- Affectivity dates
- Performance characteristics / KPIs

SOA Lifecycle – Model, Build, Run, Manage

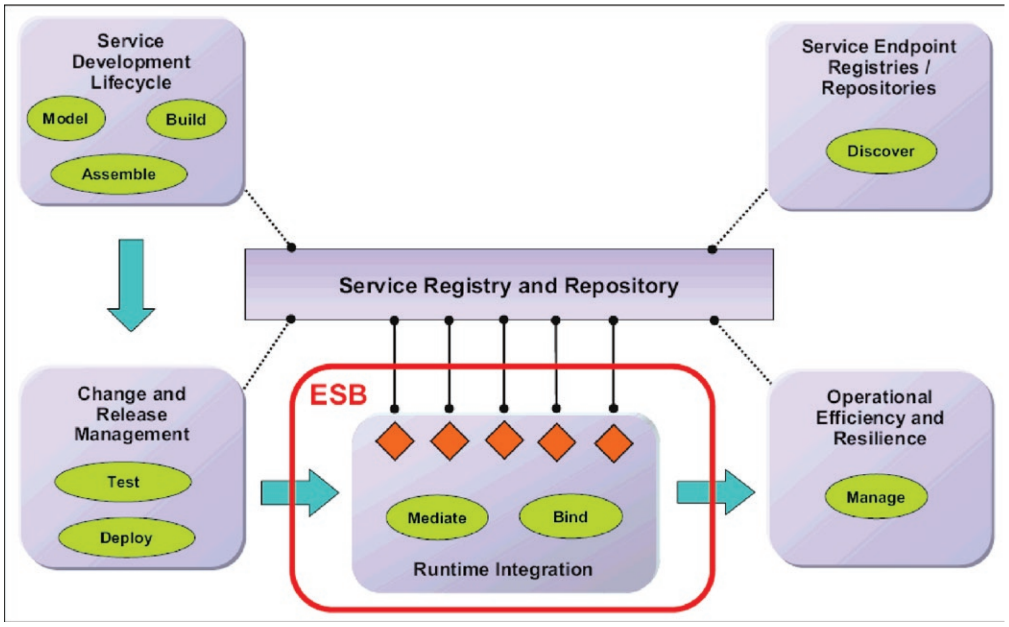
Model

With the emergence of SOA as a leading architecture for distributed systems, the need to develop a programming model to help vendors achieve this vision in a consistent fashion has arisen. IBM, BEA, Oracle, SAP AG and others have collaborated on such a model. Service Component Architecture (SCA), a key aspect of this model, exposes business logic offered through service-oriented interfaces and consumes functionality of other components (service references) through similar interfaces. SCA divides the building of an SOA application into two steps:

1. Implement service components that provide services and also consume services

2. Assembly of components through "wiring" references of services to their implementations

Figure 13: Service Registry Example


Service Development Lifecycle
Model
Build
Assemble
Service Endpoint Registries / Repositories
Discover
Service Registry and Repository
Change and Release Management
Test
Deploy
ESB
Mediate
Bind
Runtime Integration
Operational Efficiency and Resilience
Manage

SCA uses a minimum of APIs and is implemented in multiple programming languages such as Java, C++, COBOL, BPEL, XSLT, SQL and XQuery. C# is not currently supported by SCA (if you are using C# you should use the WCF framework). SCA supports synchronous-asynchronous and request-response, as well as message-oriented communication styles. Interface bindings as well as quality of service (QOS) attributes are handled declaratively, independent of the service implementation. Interface bindings can be messaging, web services or CORBA IIOP. QOS addresses security, reliable messaging and transactional capability.

In Figure 14, Service Component Architecture (SCA) Example, SCA components (A, B) are shown to each consume services (incoming arrow) and then expose their calling interface (outgoing arrow) to be consumed. The components A and B are wired together to create a composite application (Y) and configured by mapping to their reference implementations.

Service data objects (SDOs) are another element of the programming model that work hand-in-hand with SCA. SDOs permit users to access data residing in multiple locations and to then format the data in a common way using a defined application programming interface (API). An SDO represents the data flowing between services in the SCA (and BPEL) environment and provides a common interface to data like SCA does to services. As shown in Figure 15, Service Data Objects (SDO) Example, SDOs support a data graph as a container with a tree of data objects. Each object contains metadata about its type as well as its value. The SDO also maintains a summary of changes made to the object before it is synchronized with its data store.

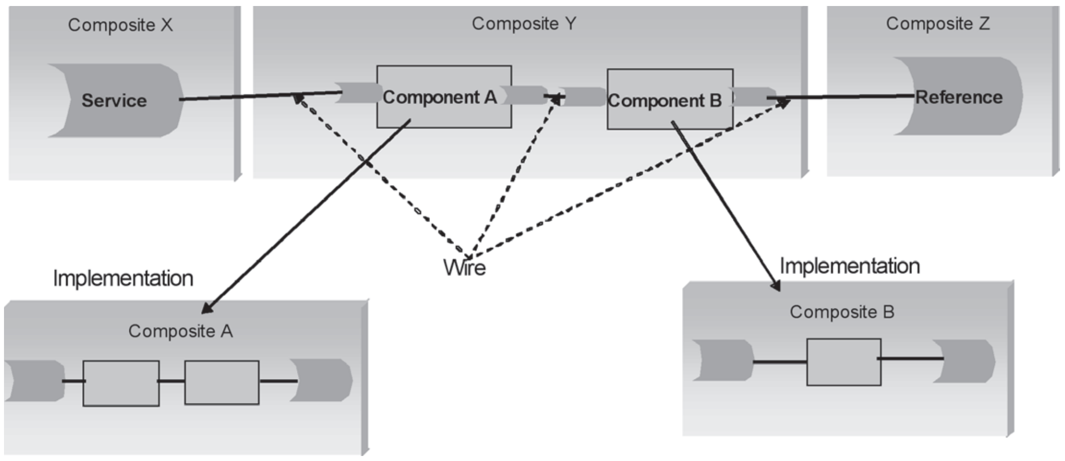


Figure 14: Service Component Architecture (SCA) Example

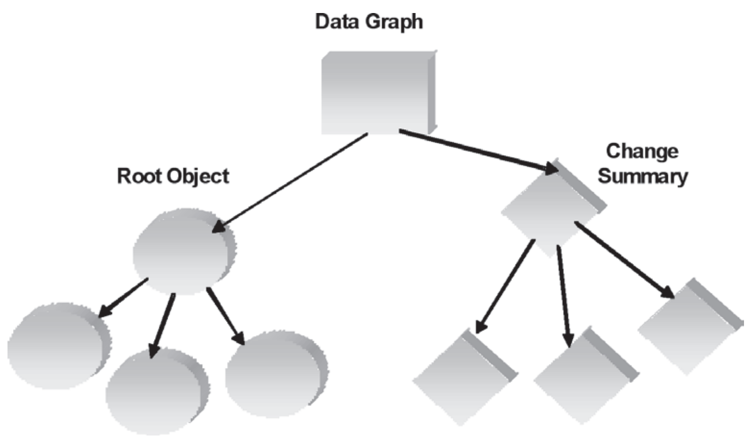
Source: http://www.ibm.com/developerworks/library/specification/ws-sca/

SOA systems are configurable in such a way that user defined SDOs are supported, and recognizable by the ESB mediation layer via the metadata configuration. SDO data mediators, not in the current standard but supported by some vendors, are responsible for creating SDOs from data sources and for updating those data sources with changes detailed in the SDO change summary. JDBC, XML and EJB are some examples of data mediators that have been implemented by some vendors. As you would expect, data mediators and SDOs reduce the programming effort and present an interface to access data, regardless of source, in a consistent and uniform fashion.

Another aspect of the SOA model is the capability to choreograph SOA components into larger composite applications. This is sometimes known as "programming in the large" and Business Process Execution Language (BPEL) is the accepted standard to accomplish this task. BPEL is an XML-based language used to define business processes that supports both short (micro) and long running (macro) life cycle models. A BPEL process invokes web service operations-based upon the WSDL 1.1 standard. BPEL provides capability to manipulate both process and control data (i.e. SDO). The BPEL language supports programming constructs like looping, switch/case statements and sequential and/or parallel path processing. BPEL also provides transactional and compensation support for the business processes. The transactional support is built upon some of the WS-I standards (i.e. WS-Coordination) discussed later in this paper. Some web service entities participating in the process flow may not be able to participate in a two-phase commit (2PC) transactional flow and, thus, are candidates for compensation which is realized as a sub-flow in the BPEL process.

Figure 15: Service Data Objects (SDO) Example

http://www.ibm.com/developer-works/java/library/j-sdo/



The web service components that are called in a BPEL process can be realized on local or remote systems, and in any language that supports the web service and WSDL standards. This makes BPEL an excellent vehicle to integrate and assemble functionality provided by a number of different vendor products and developer skill sets. BPEL processes themselves can be externalized via a web service interface so that they can be consumed in other business processes as a callable service. This allows the user to hide complexity, by encapsulating a complex business process into a higher level service which can then be exposed on the production workbench.

Build

Programming environments provide a capability for creating software components that are deployed and run on vendor-specific runtime servers. Most programming environments provide either declarative transaction and web service support (e.g. .NET), or they provide wizards and configuration parameters (e.g. J2EE with deployment descriptors, etc.) accomplishing the same purpose. For composite applications choreographed with BPEL, the environments provide tooling that will translate a flow of services, normally represented graphically, into a BPEL definition file understood by the corresponding runtime engine. The BPEL standard is abstract and makes no requirements as to how the BPEL process will be implemented on a particular vendor's hardware. Each vendor takes the BPEL definition and extends it to take full advantage of the vendor's platform features to realize an executable process. Because of this, it is necessary to use tooling that supports a specific vendor runtime to generate and deploy an executable BPEL process for that environment.

When using a services oriented architecture in the manufacturing space, one can view the architecture as having an 'extended ESB' or Manufacturing Services Bus (MSB). The MSB provides a superset of functionality of the ESB previously described. The MSB adds the following functionality on top of the base ESB functions:

- Build time workbench (aka production workbench) to model process events and corresponding actions in the context of a manufacturing line configuration

- Device access services to facilitate communication between the manufacturing devices and the ESB
- Standards-based manufacturing services for basic MES-type functions (e.g. WIP Tracking)
- Support for applicable manufacturing integration standards (e.g. ISA-95, OAGIS®[1]).

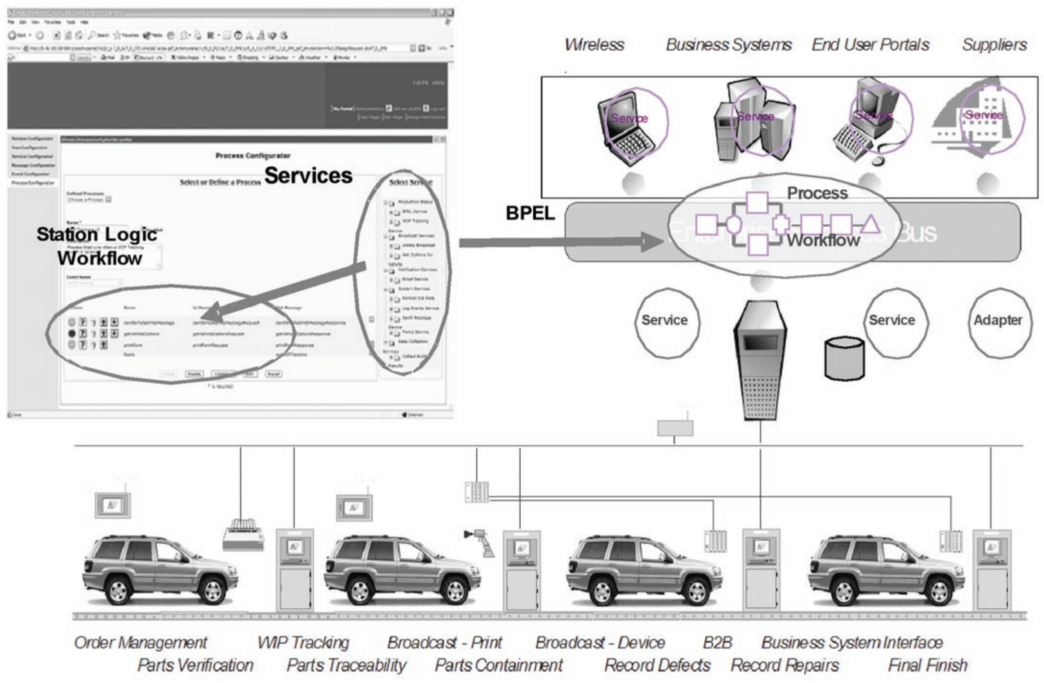
The production workbench is the key to making the MSB customizable and configurable by nonprogrammers. With the production workbench, one can build a line configuration consisting of the following:

- layout of the manufacturing line
- business processes that run to support that manufacturing line
- events that trigger the business processes
- linkage between manufacturing line process points (stations) and events
- mappings between the data in the events and the services within the business processes that those events trigger

Figure 16 shows how the workbench is used to ultimately create a composite application, or business process, of available services. The workbench allows the user to assemble available services together to form a composite application. To support a specific plant's workflow, the composite application can be deployed to a runtime instance and executed as a result of a particular event coming into the ESB.

After building the line configuration, the workbench is used to generate and deploy the runtime artifacts (e.g. executable code and configuration settings) necessary to realize that manufacturing line configuration. The standard output from the workbench is BPEL, which can be transformed, if necessary, into a workflow representation for any supported runtime engine. After the model of the line configuration is deployed to the runtime engine, events can be processed against it.

Figure 16: Constructing Composite Applications with Services and BPEL



Run

The key component to any SOA runtime environment is the Enterprise Services Bus (ESB). Business events arrive at the ESB in many forms and result in triggering business logic. As described above, the ESB is responsible for routing, converting, transforming and handling business events. We also introduced, above, the concept of an extended ESB (known as the MSB) for the manufacturing industry, which adds functionality to the base ESB applicable to manufacturing environments.

Device access services within the MSB pass data to and from the manufacturing device layer (see Figure 17). The manufacturing devices send events, via the device services, to the ESB. The ESB mediates the event from its device layer format into the common canonical format (internal format), based on the line configuration created using the production workbench (described above). Once the event is transformed into its canonical format, it can be used by the MSB components to call the appropriate business process, again, based on the line configuration. The output of the business process is mediated by the ESB into its device layer format, and sent back down to the device layer via the device access services in the MSB.

Figure 17: Manufacturing Services Bus

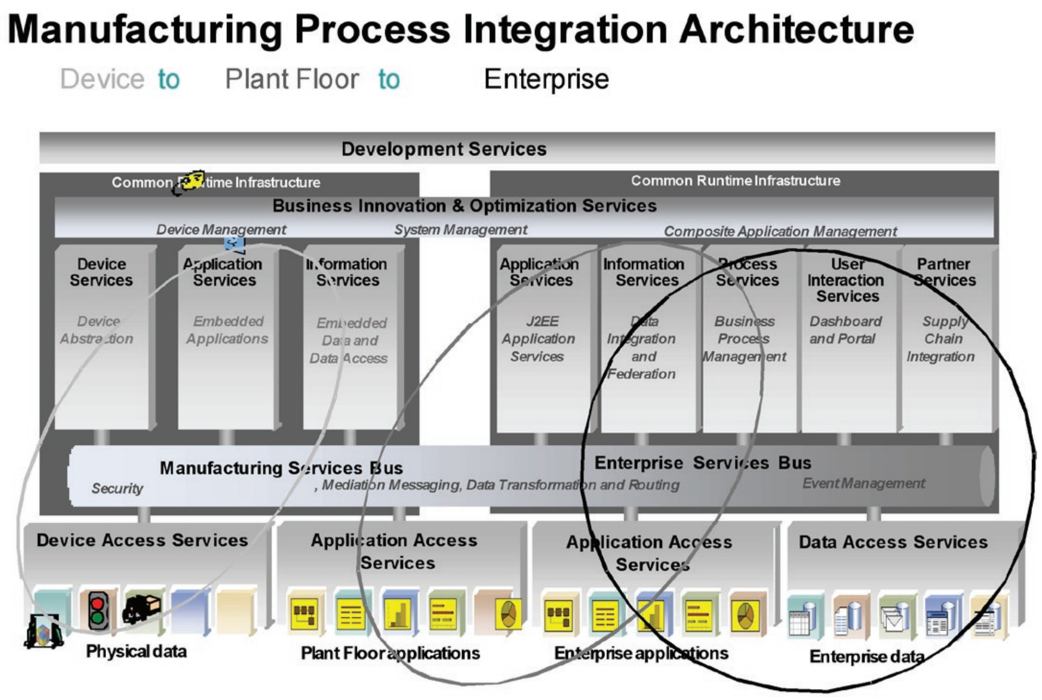

Manufacturing Process Integration Architecture
Device to Plant Floor to Enterprise
Development Services
Common Runtime Infrastructure
Common Runtime Infrastructure
Business Innovation & Optimization Services
Device Management
System Management
Composite Application Management
Device Services
Device Abstraction
Application Services
Embedded Applications
Information Services
Embedded Data and Data Access
Application Services
J2EE Application Services
Information Services
Data Integration and Federation
Process Services
Business Process Management
User Interaction Services
Dashboard and Portal
Partner Services
Supply Chain Integration
Manufacturing Services Bus
Enterprise Services Bus
Security
, Mediation Messaging, Data Transformation and Routing
Event Management
Device Access Services
Application Access Services
Application Access Services
Data Access Services
Physical data
Plant Floor applications
Enterprise applications
Enterprise data

Figure 17 depicts the MSB architecture. It is important to note that events can come from any layer of the enterprise. Regardless of the event source, or location, the event processing pattern is the same. Events enter the ESB in any format that is supported by the event client. The event is then mediated, using the line configuration metadata, into the common canonical, MSB, format. Once the event is transformed, the MSB chooses the appropriate business process to execute, by inspecting, processing, and comparing the event payload with the line configuration metadata. After the business process is executed, the output is then mediated from the MSB common canonical format back to the original format that is supported by the event client (or other output consumer) and sent to the designated output destination.

If the business processes, or more generally, the line configuration, change in any way, one can update the line configuration in the production workbench and re-deploy that configuration to the runtime instance. The goal of such a workbench is that this require no coding or IT development skills. The new line configuration will then be used to process subsequent events.

Manage

It is vital to be able to monitor and manage the event traffic within the manufacturing IT infrastructure at all times. The MSB is responsible for logging all event traffic, keeping track of event times, results and performance metrics. All the event data in the system is logged using an approved standard; for example Common Based Events (CBEs) (now part of the OASIS WSDM standard: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm). Using an approved standard event format will allow for the flexibility to use any monitoring tool that recognizes that standard.

Once the event traffic is logged, many options are available for viewing that event data. Portlet technologies can be used to display the event data in a portal. Other web-based technologies can be used to visualize the data in a web-browser. Additional vendor-specific monitoring tools that support the CBE format can also be used to visualize the event traffic.

Service Registry/Repository

As more services are created, a service registry becomes an important build time and governance tool, which facilitates the sharing of information about available services, enabling re-use throughout the organization.

The registry may also be used at run time. For example, WS-Addressing may be used along with a registry to enable dynamic change of endpoints when needed (e.g. if a provider is offline). However, the performance implications of this must be considered when considering doing this at the MES or control layers which have strict performance requirements.

USING STANDARDS TO GET THE FULL SOA BENEFIT

The evolution from point-to-point to hub-and-spoke and then to a services oriented architecture has provided incremental efficiencies in regards to the way that applications integrate with each other. There are also additional efficiencies in the way that applications are able to adapt to business change. The 'Service Oriented Architecture – an Overview' section provides details on these different architectures. While, the hub-and-spoke and service bus architectures are more efficient than a point-to-point architecture, there are other areas to be addressed. The use of standards significantly increases the technology change efficiencies. For example:

- Communication protocol standards allow easy interconnection between different buses (or hubs).
- J2EE standards allow interoperability between different application servers
- Messaging content (semantic) standards decrease the number of transformations required as messages flow on the bus. Mediations at the bus allow for transformation to a canonical format, reducing the number of transformations required between applications. The use of a canonical format may be considered a standard, but if the canonical format used is proprietary, then further mediation is required for any external communications. The use of industry standards for the canonical formats makes it more likely that no mediation will be needed on the content when connecting to a new application or a new external gateway, given the possibility that the new application or gateway may already be using the industry standard. The use of industry standards for the canonical format will also decrease the need for new mediations due to market or business model changes.

The use of a standard for content by the application itself decreases the number of mediations required on the bus, and thus improves performance.

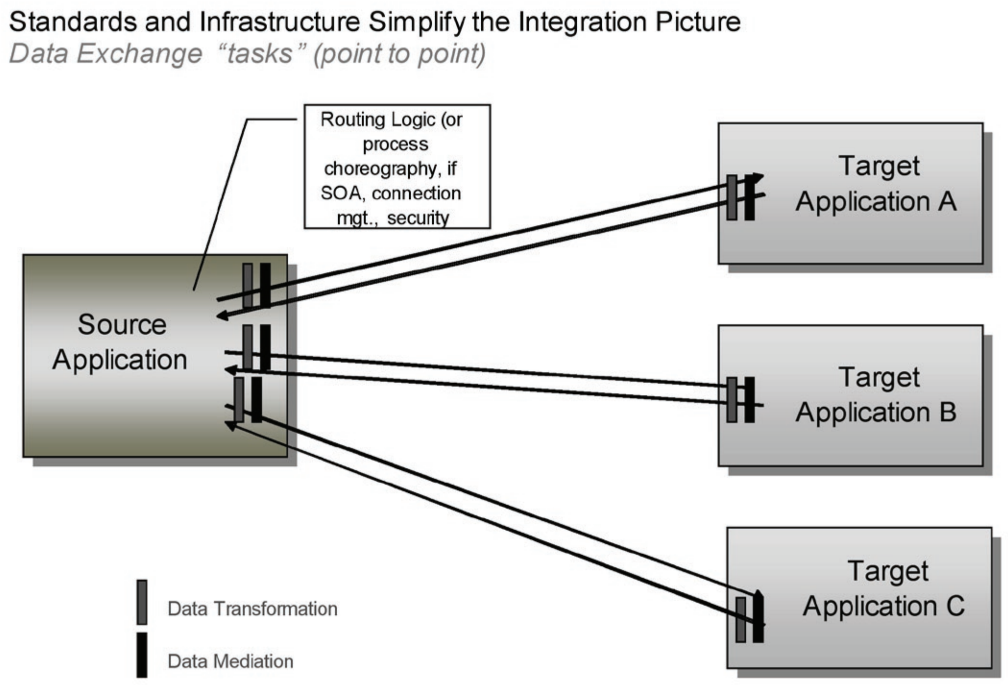
Using an Enterprise Service Bus (ESB) without message content standards can be considered to be analogous to doing point-to-point integration with new technology (with regards to the number of transformations required to make any changes to the endpoints). Considering the basic tasks needed to exchange information between systems, each exchange requires the following:

1. Routing decisions, perhaps rule-based, on where the information should be sent
2. Data transformation from the source format to that required by the target application interface
3. Possibly a different communications protocol for each exchange as well, depending upon the target application's capabilities
4. Mediation of data content (units of measure, for example, might need to be converted from "KG" at the source to "LB" at the target).

Additional considerations for information exchange include security, process choreography, monitoring, etc.

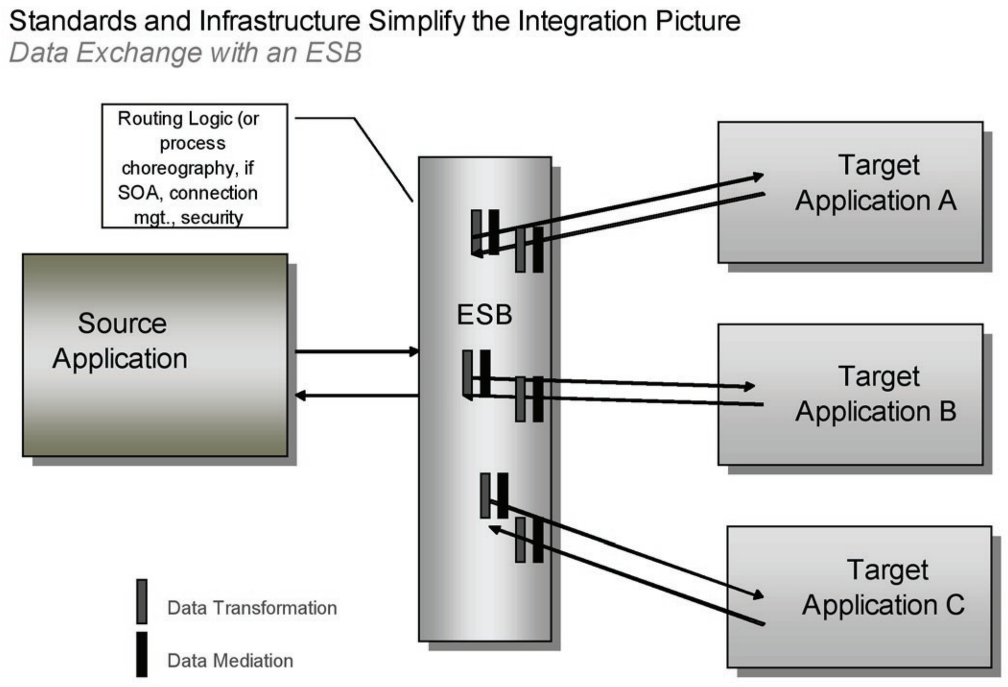
A point-to-point configuration requires transformations and mediations at each endpoint, as depicted in Figure 18, Point-to-Point Mediations.

Figure 18: Point-to-Point Mediations


Standards and Infrastructure Simplify the Integration Picture
Data Exchange "tasks" (point to point)
Routing Logic (or process choreography, if SOA, connection mgt., security
Source Application
Target Application A
Target Application B
Target Application C
Data Transformation
Data Mediation

The addition of a message bus does simplify the architecture by implementing functionality such as routing logic, security, etc. within a centralized location. Likewise, both source and target applications are simplified by not having to handle data transformations or mediations. However, as shown in Figure 19, Mediations on the Bus, the transformations and mediations must still be performed within the infrastructure, if the data content has not been standardized.

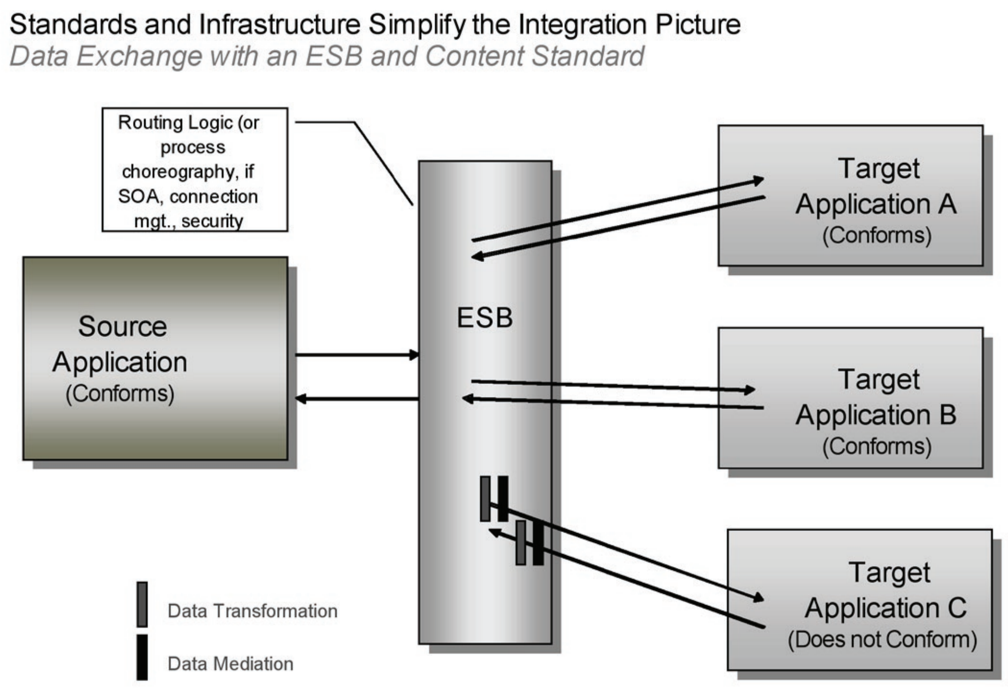
Figure 19: Mediations on the Bus


Standards and Infrastructure Simplify the Integration Picture
Data Exchange with an ESB
Routing Logic (or process choreography, if SOA, connection mgt., security
Source Application
ESB
Target Application A
Target Application B
Target Application C
Data Transformation
Data Mediation

As applications adopt content standards, the data transformations and mediations can gradually be eliminated, thus further simplifying the task of maintaining the integration. Note: Configurable middleware supports this happening in an evolutionary fashion.

Figure 20, Reduced Mediations with Data Content Standards, shows an example of this. Note that only the applications not conforming to the content standard requires transformations/mediations:

Figure 20: Reduced Mediations with Data Content Standards


Standards and Infrastructure Simplify the Integration Picture
Data Exchange with an ESB and Content Standard
Routing Logic (or process choreography, if SOA, connection mgt., security
Source Application (Conforms)
ESB
Target Application A (Conforms)
Target Application B (Conforms)
Target Application C (Does not Conform)
Data Transformation
Data Mediation

If the content standard adopted is an industry standard, this allows integration to partners with fewer transformations or mediations and allows "plug in" of commercial off-the-shelf (COTS) applications that support that standard.

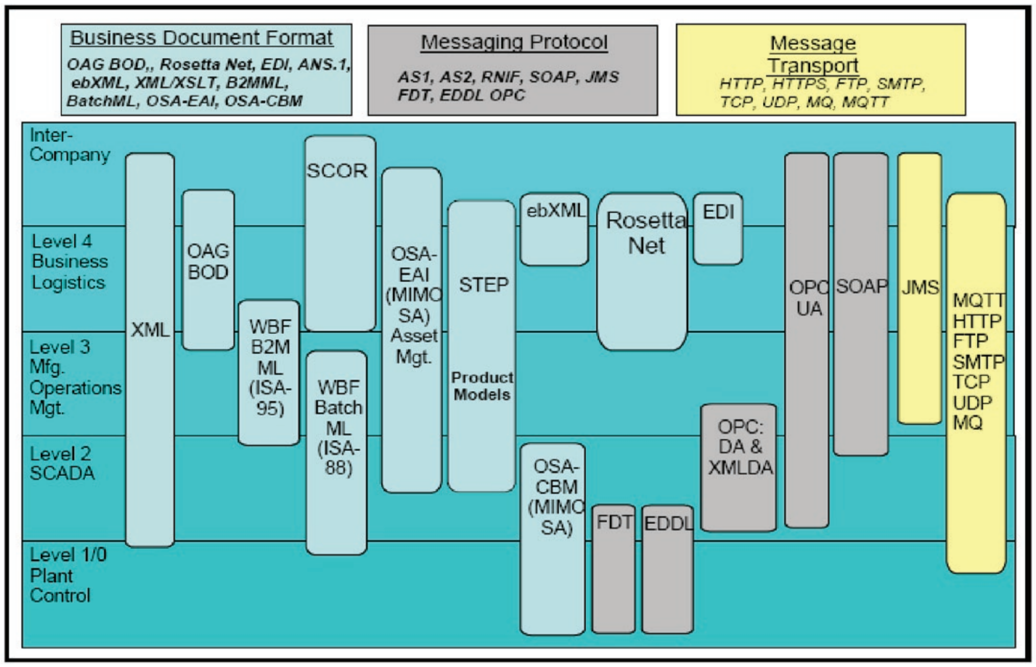
Message Content and Format Standards

Whether using a hub-and-spoke or SOA architecture, using a common canonical data format for the exchange of data reduces the number of data transformations necessary as multiple applications communicate with each other. There are multiple sets of standards for integration at different levels of manufacturing operations. These standards can be a logical starting point for the canonical data format to be used on an ESB (or a hub). If the applications use the same canonical standards for communication, the need for mediations and transformations is reduced.

The standard(s) to start with vary based upon individual needs including the process level being used. Regardless of the standard(s) chosen, some level of customization is likely to be required based on customer need. This may entail choosing a subset of standard attributes or extending the standard objects with additional required attributes.

Figure 21, Manufacturing Integration Standards – An Overview, gives a sample of the various manufacturing standards in use today: For more details on some of these standards and how they fit in a manufacturing environment, please refer to the MESA White Paper 26, "Related Manufacturing Integration Standards, A Survey (http://www.mesa.org/knowledge-base/details.php?id=75).

Figure 21: Manufacturing Integration Standards—An Overview


Business Document Format
OAG BOD,, Rosetta Net, EDI, ANS.1, ebXML, XML/XSLT, B2MML, BatchML, OSA-EAI, OSA-CBM
Messaging Protocol
AS1, AS2, RNIF, SOAP, JMS FDT, EDDL OPC
Message Transport
HTTP, HTTPS, FTP, SMTP, TCP, UDP, MQ, MQTT
Inter-Company
Level 4 Business Logistics
Level 3 Mfg. Operations Mgt.
Level 2 SCADA
Level 1/0 Plant Control
XML
OAG BOD
WBF B2M ML (ISA-95)
SCOR
WBF Batch ML (ISA-88)
OSA-EAI (MIMOSA) Asset Mgt.
STEP
Product Models
ebXML
Rosetta Net
EDI
OSA-CBM (MIMOSA)
FDT
EDDL
OPC: DA & XMLDA
OPC UA
SOAP
JMS
MQTT HTTP FTP SMTP TCP UDP MQ

XML

The Extensible Markup Language (XML) is an open standard, general purpose markup language. XML is human and machine readable and is self-documenting. It serves as the basis for most of the following standards.

XML Schema

XML Schema is a schema language that has been approved by W3C which can be used to define the rules for message formats to be used for exchange of data between MES, ERP and other systems. The XML Schema for these standards describes the structure of the XML document and is an XML alternative to Document Type Definitions (DTD). The XML Schema language instance is referred to as XML Schema Definition (XSD) describing the format and structure of the document. The schema becomes an interface agreement between the document producer and consumer and insures interoperability.

ISA-95 and ISA-88

Starting with ISA-88, Batch Control, in 1986 and followed by ISA-95 (AKA: IEC®1 62264) in 1995, the ISA®1 (www.isa.org) has been progressively addressing integration of business and manufacturing applications. In particular, the ISA-95 standard has adopted:

1. A functional and physical model of an application hierarchy
2. Terminology for manufacturing operations management (MOM)
3. Methodology to describe information exchanges

4. A common data object model
5. Activity model for MOM

See Figure 1 for a depiction of the ISA-95 domain hierarchy.

B2MML; BatchML

Business to Manufacturing Markup Language (B2MML) is an XML implementation of ISA-95 and **Batch Markup Language (BatchML)** is an XML implementation of ISA-88. Both are developed and published by the WBF organization (formerly the World Batch Forum).

OAGIS

The Open Applications Group Integration Specification, OAGIS, has established a common canonical standard for the information to be exchanged between applications, as well as a standard way of "packaging" that information. OAGIS defines Business Object Documents (BOD) with standard structures and headers, with a body unique to the BOD. B2MML and OAGIS BODs are both extensible, albeit in different ways.

MIMOSA

The Open System Architecture for Enterprise Application Integration (OSA-EAI) architecture is a specification published by the Machinery Information Management Open Systems Alliance (MIMOSA) organization. MIMOSA publishes XML-based specifications for Enterprise Application Integration (EAI) and Condition-based Maintenance (CBM), including detailed models for assets and equipment.

STEP

The **Standard for the Exchange of Product (STEP)** Model Data is an ISO standard (ISO 10303) describing how to exchange digital product information, including Computer Aided Design (CAD), Computer Aided Engineering (CAE), and Computer Aided Manufacturing (CAM) systems.

Service Platform

J2EE (Java EE)

Java 2 Platform, Enterprise Edition (J2EE) provides a standard platform for Java application server programming. J2EE includes specifications such as JMS, web services, XML, Enterprise JavaBeans, servlets, portlets and JavaServer pages. J2EE provides the capability to deploy portable, scalable, fault-tolerant, distributed, multi-tier Java applications.

The original J2EE specification was developed by Sun Microsystems. Later versions of the specification were developed using the Java Community Process.

Note that the name has changed to Java Platform Enterprise Edition (Java EE) with version 1.5.

.NET and integrating Microsoft applications

One of the goals of SOA is the ability to integrate and consume existing application functionality without the cost of rewriting and replacing code where there is a current significant investment. Another concern of some companies is the ability to leverage different skill sets within the company or with a partner/vendor. SOA and web services allow companies to successfully address both these concerns. One example of this success is the interoperability between J2EE and .NET applications and in particular between Microsoft and J2EE vendor web service products. As mentioned previously, while there are differences in the approaches these technologies leverage to provide SOA functionality, there are a number of standards that both embrace that make interoperability possible. Some of these commonly accepted standards are addressed below.

Interoperability between .NET and J2EE

J2EE and .NET applications can be integrated at a number of different levels. Two papers, "Application Interoperability: Microsoft .NET and J2ee" (http://download.microsoft.com/download/7/2/6/7269f183-639a-4e99-bd84-cc3e6515af86/PnP_J2EE_Interop_V1.pdf) and "WebSphere and .NET Coexistence" (http://www.redbooks.ibm.com/abstracts/sg247027.html?Open) , address different architectures and integration patterns including a web service interface. This paper advocates and investigates only the web service option. The ability of vendor technologies to interoperate at a web service level is dependent upon their compliance to standards. The WS-I standards organization has created a series of standards that address different aspects of web service interoperability. The extent that these standards explicitly define and address compliance to the specifications will determine the level of interoperability achieved. Those areas that tend to give vendors room in how they implement the standards increase the possibility of interoperability problems. In this section, we highlight a few areas where standards provide vendors "room for interpretation" that can result in interoperability issues. While there are many web service standards, we will try to touch on and highlight a few of the major ones. Not all the standards are implemented by all vendors (e.g. WS-ReliableMessaging). An illustration of some of the profiles is shown in Figure 22. Figure 22, as well as some of the information used in this section, can be found in IBM Redbook SG24-6395 (http://www.redbooks.ibm.com/abstracts/sg246395.html), which is an excellent paper on WebSphere®[1] and .NET interoperability with Web Services.

WS-I (Web Services Interoperability) Profiles

The strength of SOA is its ability to access and invoke functionality in a variety of operating environments and programming languages. The requirement to ensure the inoperability of these various environments leads to the rise of web services interoperability standards. These standards are generally known as the WS-I specifications. These specifications address various characteristics of the interactions that occur during normal processing. Strict compliance to these standards leads to a higher degree of interoperability between various vendors and reduces the effort of integration while raising the level of confidence in a distributed environment scenario. The WS-I standards are being created by the WS-I organization, an open industry organization (http://www.ws-i.org/) committed to web service operability across operating systems and programming languages. This organization has been embraced by several other standards bodies including AIAG, OAGIS, OPC Foundation, RosettaNet, MIMOSA, etc. This support has lead to its wide acceptance in the development community.

As shown in Figure 22, WS-I Profile Stack, this standard encompasses a number of areas. There are several WS-I Profiles: Basic, Secure, and Reliable Secure.

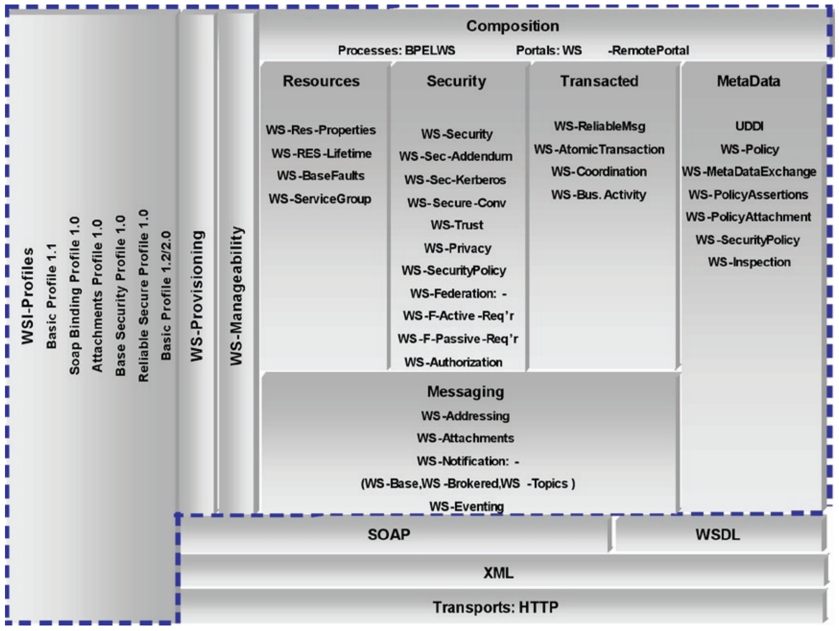


Figure 22: Web Services Support for .Net/J2EE Interoperability

Basic Profile

WS-I Basic Profile (1.0) (http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html) is a set of non-proprietary Web service specifications. The specification places requirements on:

1. Message – protocol elements exchanged (SOAP/HTTP messages)
2. Description – types, messages, and interface (WSDL)
3. RegData – registration and discovery (UDDI).

Basic Profile requires specific levels of standards like WSDL 1.1, SOAP 1.1, XML 1.0, HTTP 1.1, SSL Version 3.0, etc. Basic Profile requirements are specific enough that interoperability between .NET and J2EE applications can be expected in most cases. However, the following are some guidelines that should be followed to ensure the best possibility of success.

- It is best to use document/literal in WSDL definitions with J2EE and .NET applications.
- Developers should try to maintain unique web service names in an application and, if not possible, use unique domain name for a namespace.
- Arrays are handled slightly differently in .NET and J2EE. IBM WebSphere® Application Server, to name one J2EE example, may return null as a result of a web service invocation which may cause a .NET application problems. Given that, in order to achieve interoperability between .NET and J2EE, it is recommended to return an empty array instead of a null value.
- When naming services, developers should follow Java naming conventions for operations with no special characters like underline ("_").
- Avoid usage of complex data types in web service interfaces if possible, simple is better for interoperation. Appendix E contains a partial list of those data types and mappings in J2EE and .NET.

Secure Profile

The WS-I Security profile described at http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2004-05-12.html covers authentication, authorization, message integrity and confidentiality. Authentication methods can be achieved through use of username token, plain text password or password digest. Transport Layer Security (TLS) and its predecessor Secure Sockets Layer (SSL) are cryptographic protocols (encryption) and have known security issues which are better addressed in WS-I Security, namely:

1. TLS does not allow selective encryption of message contents, it is all or nothing and can result in high overhead and degraded performance
2. TLS only guarantees message security while the message is "on the wire", so, as an example, if the message has to sit in a queue waiting for an application receiver it is unencrypted at that time.

WS-I Security addresses these deficiencies in that it permits encryption of only those portions of the message that are sensitive and it moves the secure boundary to the application. The selective encryption capability avoids the overhead of encrypting the full message content and allows routing information to remain unencrypted so it can pass through firewalls.

Reliable Secure Profile

WS-ReliableMessaging (http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf) is a specification proposal for reliable interaction with four basic delivery assurance patterns:

1. Messages are delivered only once.
2. Messages are delivered at least once.
3. Messages are delivered at most once.
4. Messages are delivered in the order sent.

The WS-ReliableMessaging standard is now agreed upon http://www.oasis-open.org/news/oasis-news-2007-06-21.php. SOAP/JMS and SOAP/MQ are nonstandard implementations that attempt to achieve reliable messaging, but because they are implemented using a specific vendor's transport mechanism they do not provide true vendor interoperability.

Additional Web Service Specifications

WS-Addressing

Another profile, WS-Addressing (http://www.w3.org/TR/2005/CR-ws-addr-core-20050817/) was submitted as a standard by Microsoft, IBM, BEA and others and has recently been approved by W3C. It introduces two constructs, endpoint references and message information headers. These constructs enable dynamic change of endpoints when needed (e.g. if a provider is offline), as well as asynchronous and stateful communication. Most typical web service requests are either request-response or datagram (send a request and don't wait for a reply). With WS-Addressing message header information, developers can enable long running asynchronous web service requests where responses are redirected based on the SOAP message header information. Header information allows the receiver of the asynchronous response to correlate the response to an original request and to enable the stateful behavior of a series of two or more web service requests between a client and provider.

WS-AT

WS-AT, Atomic Transactions (http://schemas.xmlsoap.org/ws/2004/10/wsat/) covers the coordination of multiple web service calls in a homogeneous or heterogeneous environment in a single unit of work (UOW). This is a powerful capability when you need to coordinate a number of web service calls into a single workflow as is supported by the BPEL standard. As an example, when a web service call fails in a workflow you may wish to roll back a previous web service call that, for example, might have updated a database.. The transaction coordinator of the two-phase commit (2PC) UOW can be either in .NET or J2EE. Since it is a 2PC, it should be restricted to short running processes. In a J2EE environment the steps necessary to make a web service capable of participating in a WS-AT transaction can be as simple as changing a deployment descriptor value with no required coding. A declarative style is used for marking transactional objects in .NET. These transactional directives can be used to declare that a .NET web service participates in a global transaction.

Transactional Support

There are two well known specifications which define how transactional support is provided for workflows consisting of web services. WS-AT is a specification covering the coordination of short lived workflows (of web services) within a UOW known as an atomic transaction. WS-BA (Business Agreement), conversely, is a specification for long lived scalable transactions (i.e. an interruptible workflow). These WS-Transactions are an extension of WS-Coordination which addresses composition of multiple web services into a single workflow. WS-Coordination is covered in detail at http://docs.oasis-open.org/ws-tx/wscoor/2006/06.

The WS-I profiles provide a detailed specification of an environment that allows a diverse set of vendors and products to interact in a reliable fashion as long as the standard is clearly defined and the vendor has faithfully implemented that standard. In areas where the standards are vague or allow interpretation, interoperability becomes a problem. The WS-I profiles are still a work in progress and as they mature and gain acceptance we should expect great strides in interoperability.

It's fashionable to say that SOA isn't about technology, but the fact that there is WS-I and interoperability between .NET and J2EE (and it's not that hard to implement the WS-* profiles in other environments) is a major development in the history of computing. As a result of this progress, implementing SOA is now much more affordable.

BPEL

The Business Process Execution Language for Web Services (WS-BPEL 2.0) specification (add reference) provides a means to formally specify business processes and their interaction with Web Services and other protocols.

WS-BPEL extends the Web Services interaction model and enables it to support business transactions. Contributors to the WS-BPEL standard include:

- IBM
- BEA Systems
- Microsoft
- SAP AG
- Siebel Systems

WS-BPEL allows services to be 'choreographed' together into a business process. The services may be local or remote, and may be implemented in various languages, allowing interoperability across heterogeneous environments. For example, legacy applications can be "wrapped" in a Web Service for inclusion in a BPEL business process. This loose coupling of services makes it easy to upgrade or replace a service without impacting the entire business process.

The business services described by a WS-BPEL process may, in turn, be exposed as a service, and used by another business process.

WS-BPEL is fully compatible with the J2EE standard. Extensions to BPEL include the ability to allow human interaction within a process flow.

OPC-UA

The OPC[1] Unified Architecture (OPC UA) is a new proposed standard from the OPC Foundation. The focus of OPC UA is enterprise integration. OPC UA is based on SOA, web services, XML-based architecture, including WS-* standards. While based on these standards, the intent is to provide a proprietary OPC binary transport for performance reasons. The OPC standards currently have different objects and methods, depending upon the type of server:

- Alarms and Events
- Data Access
- Historical Data Access
- Commands
- Complex Data

For example, every server currently has a "Browse" method, but the syntax is different for each. OPC UA proposes a unified object model that would expose a common set of variables, methods, and events across the multiple different types of existing OPC servers. The proposed stacks to be supported include .NET, Java, C/C++.

For further information, please see: http://www.opcfoundation.org

CONCLUSION

Although the technology to support SOA in manufacturing has clearly arrived, the fact is that most manufacturers have not really progressed very far on this initiative as of yet. There seems to be a general recognition that there is a need to do so, and many companies have initiatives now underway in this area, but this is going to be an evolutionary, rather than revolutionary process.

Since SOA is all about standards, it is possible to mix together offerings from one vendor with another, although like most standardization efforts, there is a bit of "wiggle room", so this might not be completely plug-and-play. The BPEL standard (WS-BPEL 2.0), for example, is still vague enough that there is no guarantee that a BPEL process created with one vendor's tools will run without "tweaking" on another vendor's BPEL runtime engine. Web Services are a bit further along in this area and are generally plug and play across vendors. Pressure from customers is forcing the issue here and these issues will be resolved.

Manufacturers can begin with moving to SOA now, and many have begun. As was previously stated, it is not necessary to do this as a step function. This can be done on a project basis, although it is important to first establish a target architecture and direction so that your projects can be planned as steps towards that eventual goal. You will also need to make some initial decisions on SOA infrastructure, although if you move incrementally you have more opportunity to evaluate approaches and implementations and to revise your plans as you go along.

Is SOA "real"? Yes, and as described in this paper, real projects are now underway as manufacturers begin to adopt this approach to bring more agility to their manufacturing systems.

Appendix A: Examples of Successful SOA Implementations

The following two examples detail real SOA implementations. These examples are not meant to be a comprehensive list of successful SOA implementations. Other examples can be added to provide additional, valid instantiations of successful SOA projects.

IBM Microelectronics Division

In 2003, IBM Global Engineering Solutions (GES) (a combination of the Microelectronics Division and Engineering & Technology Services) which is a part of the Sever and Technology Group (STG) made a request of the IT group to support a new business model. Specifically, they requested support for manufacturing at suppliers in the Asia Pacific (AP) region in order to provide additional capacity to the IBM manufacturing base. Further, a challenge was provided that the outsource locations must embody existing IBM manufacturing intellectual property (IP). As described in a simple manner, the engineering staff desired to view the outsource locations as transparent, being able to receive the same exact raw data and logistics information as they had from internal areas. This was a major challenge, given that there was a large volume of data, much of which was in proprietary formats.

The selected solution embraced SOA and created a message-based architecture with custom services. These services at the suppliers would provide the same IP that was used for internal sites, the services embodying the process IP. Coined as a "factory in a box," a single server was delivered directly to the supplier, containing everything required to support a particular manufacturing area. IBM GES also chose MQ Series® messaging as the interconnect methodology to link these remote servers back to a hub in IBM. WebSphere Message Broker was used to transform and route messages as the secure transport. The solution was completed by specification of a demilitarized zone (DMZ), which is separated by firewalls on each side from the internet and trusted zones of the enterprise in which this server, called Multi-Data source Integrator (MDI) would sit to satisfy both IBM and supplier security demands.

Given the deadline time constraints, there was no luxury of a pilot; it had to work on delivery. This was a bold risk to take in retrospect. However, the skill levels of the development team and careful focus on delivery of exactly what was needed mitigated the risk.

Since the initial deployment, the solution implemented has been tuned, enhanced and expanded and now supports nearly 14,000 logistics transactions per day and several gigabytes of raw data transfers. To date, we have suffered just a single outage at one supplier node lasting about eight hours that was traced to a disk drive failure that was not implemented to the RAID 5 specification.

The solution has proven to be scalable and is operating at 20% of its capacity design point per node at present. The balance of this section will provide details behind the architecture, including the reference architecture developed as a result of this initial project.

Process Modeling

To start the project, which in general is referred to internally as "Virtual Manufacturing Framework" or VMF, the team had to model the existing internal business, part of the request being to make the outsource engagements transparent to internal support groups. The knowledge of internal processes was fragmented and often had outdated documentation. There was no consolidated view of the end-to-end process and few (if any) process owners identified. Not having the luxury of time for a full and detailed process mapping, the IT resources chosen had to have very high business knowledge expertise. This provided a short cut to the required delivery time.

After the initial delivery, the team stepped back and worked on formalizing the approach, and mapping the domains to extend the architecture of the project to a more generic reference model. The following architecture was the result.

IBM GES Reference Architecture

The reference architecture shown in Figure 23 details the domains into which MD (Microelectronics Division) was broken. Each domain was chosen as a cohesive business unit, with its own applications, support and a single ground rule… "Assume the boundary chosen could, at any point, be used as a guide for potential outsource".

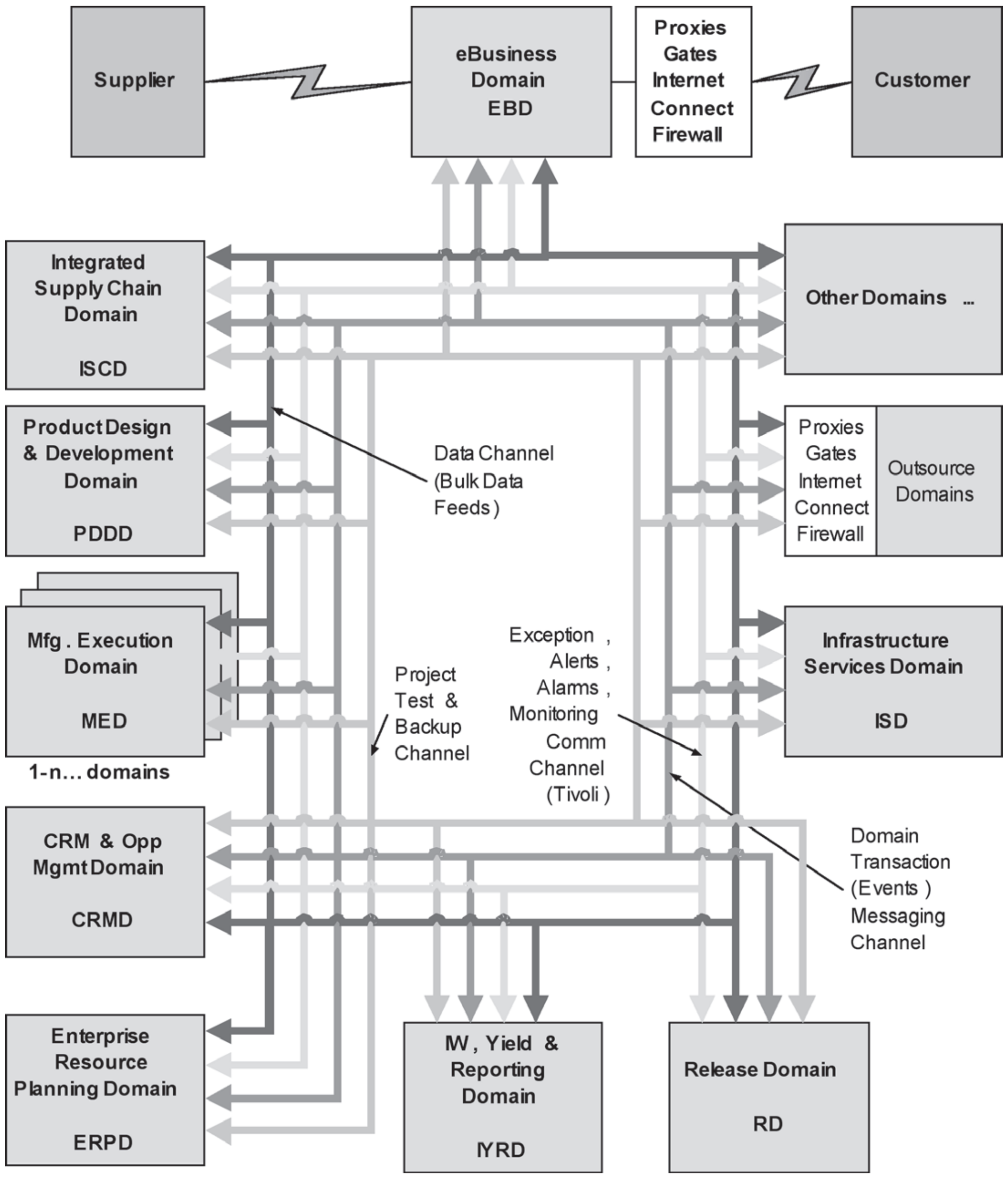
The reference architecture shows that each domain is isolated from the other domains by a messaging bus. One bus for transactions, one for large scale bulk data transfers, another for alerts and monitoring. The last, a duplicate of the transaction bus, is used for integration testing before production release. With the exception of the monitor bus, each will be implemented with MQ transactions, the monitor bus using Tivoli®1 protocols. The transaction bus uses XML with preference to standards flow as detailed by the RosettaNet™ organization for logistics flows.

The messaging infrastructure could be extended world wide, with proper proxy handling extended across the internet to suppliers.

Implementing this reference architecture required an interface component, and specifically one that could handle the message flows, transform and route as needed within the domains. Nothing existed that would meet all of the demands of each message bus in total and be extensible and secure - an invention was required. To this end, IBM created a system called MDI (originally simply dubbed as the Manufacturing Data Integrator but later re-named Multi-Data Source Integrator. The MDI systems implement SOA, and contain the services. The simplistic thought was that legacy systems could be wrapped by services in the MDI, new services could be created, and over time, only the MDI would remain as applications migrated to full

SOA. Each MDI is by today's standards and nomenclature, properly classified as a mini-ESB (Enterprise Service Bus).

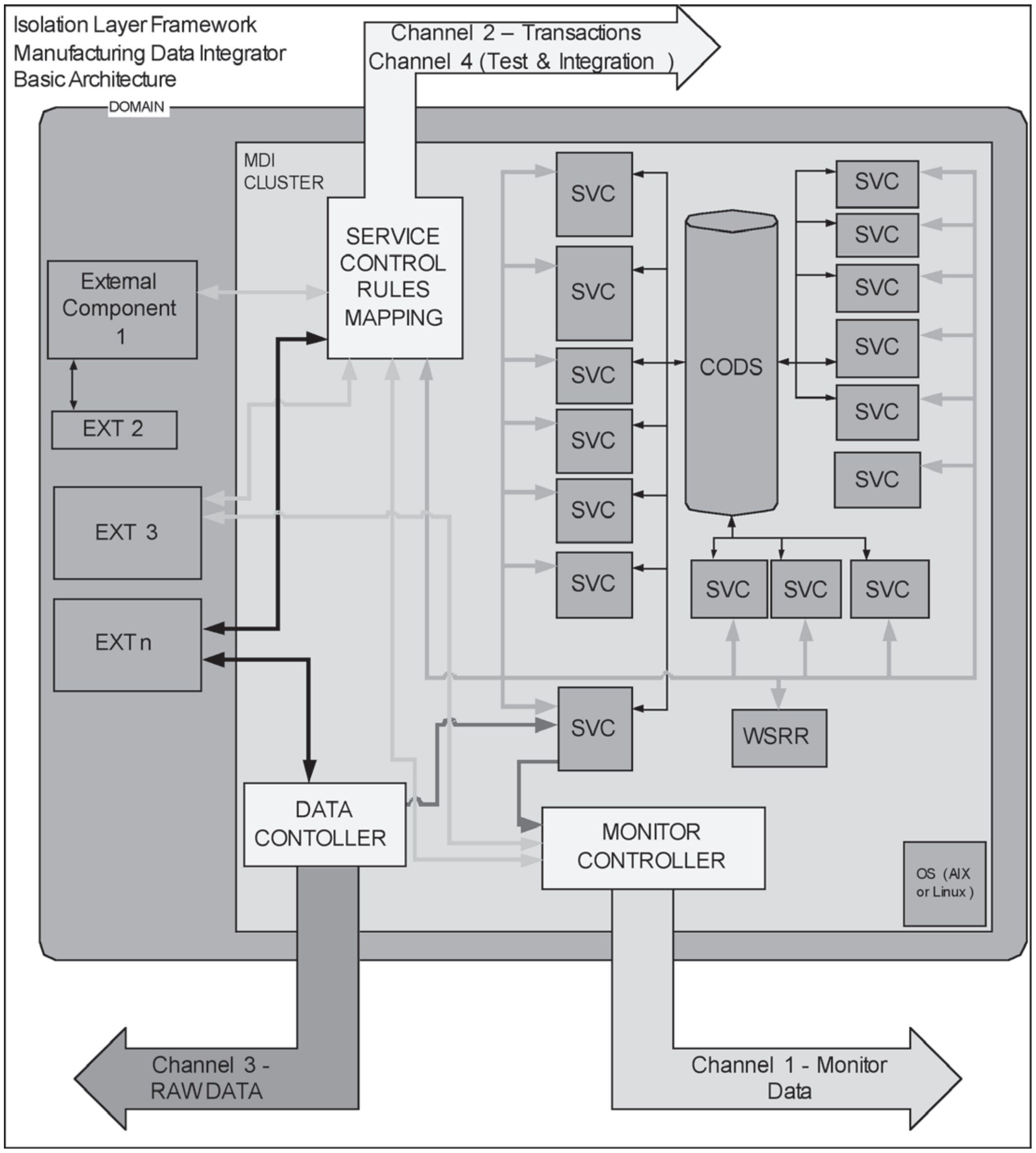
Figure 23: 2003, 2006 GES Reference Architecture V1.2


Supplier
eBusiness Domain EBD
Proxies Gates Internet Connect Firewall
Customer
Integrated Supply Chain Domain ISCD
Other Domains ...
Product Design & Development Domain PDDD
Data Channel (Bulk Data Feeds)
Proxies Gates Internet Connect Firewall
Outsource Domains
Mfg. Execution Domain MED
1-n... domains
Project Test & Backup Channel
Exception, Alerts, Alarms, Monitoring Comm Channel (Tivoli)
Infrastructure Services Domain ISD
CRM & Opp Mgmt Domain CRMD
Domain Transaction (Events) Messaging Channel
Enterprise Resource Planning Domain ERPD
IW, Yield & Reporting Domain IYRD
Release Domain RD

The MDI, the SOA Framework

The following architecture shows the components of an MDI system.

Figure 24: Multi-Data Source Integrator



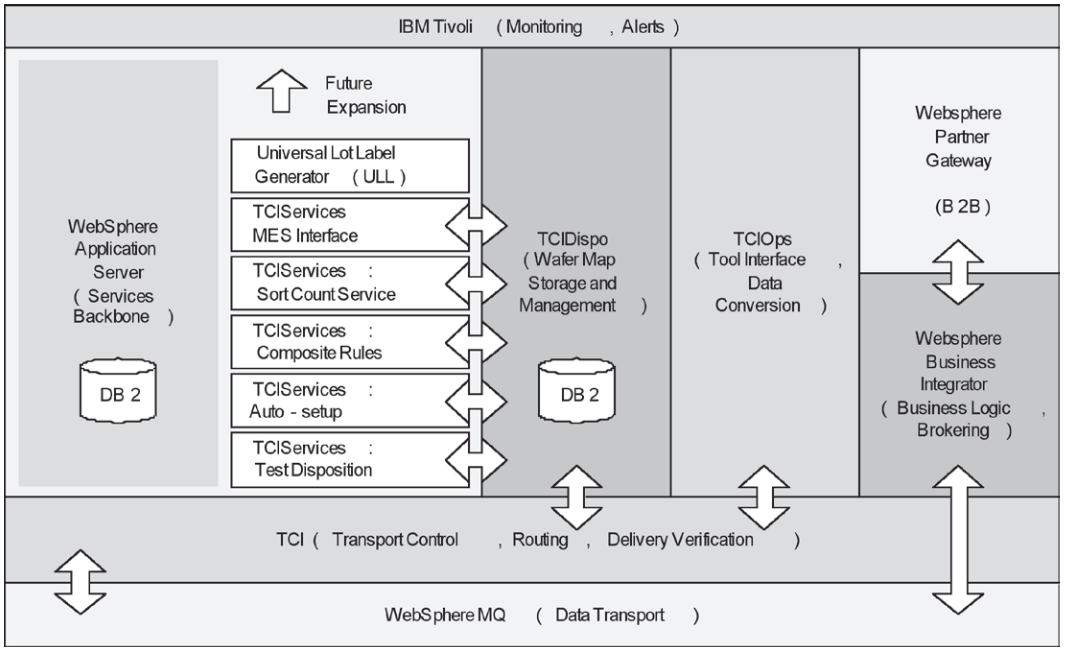
To instantiate this MDI architecture, IBM WebSphere Business Integrator with WebSphere Message Broker over WebSphere MQ were chosen. The services portion of the MDI, utilize WebSphere Application Server combined with IBM DB2® database IBM WebSphere Registry and Repository serves as the Services Registry in this deployment. The MDI platform allowed existing application developers to start porting applications (which had been vertically integrated) to two entities on the MDI; the business rules, which were now handled by workflow within WebSphere Business Integrator), and the actual business services, as coded to exist on WebSphere Application Server. The services created within the MDI, follow the

SOAP specification for interoperability. The MDI system also provides a graphical user interface (GUI) to a subset of the services for end users (supplier interfaces are not available). In this way, the services can be called manually.

This semiconductor industry implementation uses RosettaNet standards to handle logistics and selected data flows. IBM has subsequently extended these standards back into its internal sites as well.

Several unique semiconductor industry services were created that can be called. The following diagram depicts the services and the base components used in the MDI. Note, the Tool Control Infrastructure Operations (TCIOps) component is a proprietary bulk data handler and converter that can receive tooling data and normalize it to a standard format.

Figure 25: MDI Component Map


IBM Tivoli (Monitoring, Alerts)
Future Expansion
Universal Lot Label Generator (ULL)
TCIServices MES Interface
TCIServices : Sort Count Service
TCIServices : Composite Rules
TCIServices : Auto - setup
TCIServices : Test Disposition
WebSphere Application Server (Services Backbone)
DB 2
TCIDispo (Wafer Map Storage and Management)
DB 2
TCIOps (Tool Interface, Data Conversion)
Websphere Partner Gateway (B2B)
Websphere Business Integrator (Business Logic, Brokering)
TCI (Transport Control, Routing, Delivery Verification)
WebSphere MQ (Data Transport)

Tangible Business Results and Lessons Learned

No case study is complete without the requisite answers to these questions: "How did it turn out? "What would you change if you could?" "What does your current roadmap outline?"

To the first point, "How did it turn out?" – in fact better than initial thoughts. Four years into the future now and with hindsight, MD is supporting a far larger set of outsource suppliers than anticipated, with only minor tweaks. The MDI system(s) remain fully viable and handling a variety of custom work never initially foreseen (e.g. re-batching of B2B transactions). In fact, one major "win" was the divestiture of an IBM manufacturing site that was running as a domain, isolated by an MDI system. During the sale, the MDI was reconfigured to remap the incoming manufacturing start signals that previously had gone direct to the Manufacturing Execution

System (MES). Now these same signals were mapped to SAP IDOCs and delivered to the supplier SAP system. During the changeover, only a four-hour outage in manufacturing occurred as IBM dropped direct control and the supplier took over. It is important to note that the MES system is not the integration hub, but the MDI is. Separating the integration pieces from the application is an important step in a SOA.

Second, "What would you change?" Initial thoughts predicated a single MDI for each domain, even if the domain was in fact just another instance. In retrospect, the MDI can be applied to multiple domains if they are similar. In fact at a supplier location, the MDI system may be used for all of the vendors' sites as long as their network supports the connectivity specifications, and the business assumes the risk of a single point of failure (MTTF is very high even without using HA technologies).

The first deployment did not include sufficient monitors, only relying on the WebSphere MQ queue monitors. However, the team's creation of a "penalty box" provided a key component often missing in designs that involve services and message flows. Work flows call services in a synchronous fashion, with each service either transforming or consuming messages. If an error state should occur in later services, often the ability to replay initial messages to restart the flow is desired. As such, the penalty box custom component, can store, reorder and replay transactions. Any error that occurs results in the message immediately going to the penalty box, where we now have a full end-to-end support team that can disposition the error (generally this is fix the message, and replay). This has been invaluable when dealing with XML batch transactions that often do not arrive in order. The penalty box will get the out of order transactions, hold them, reorder the message flows, and then replay. All of this has been automated now. In fact, a lesson to take away is that automated error handling is a major work item not to be overlooked during initial design.

One major issue arose during the initial deployment that drove major reviews and threatened to stop the project, the complexity of the WebSphere MQ series configurations world-wide in a cluster mode hub-and-spoke topology. The added elements of WebSphere MQ proxies in our DMZ areas and redundant path requirements to ensure automated recovery added significant testing challenges. In the end, the initial architecture prevailed, but not without calling in IBM WebSphere MQ level 3 support and confirming the base infrastructure architecture was sound. In the end, the root cause was traced to a series of small configuration typos in the WebSphere MQ build. The team also found, issues with messages arriving out of order. A series of cases within the IBM WebSphere MQ / JMS environments with priority and message persistence options were identified as the root cause of this problem. A simple lesson from that is: if two technologies in use both offer the capability to prioritize flows, choose one, and stay consistent. The choice the team made was to utilize WebSphere MQ priority attributes, as there were potentially flows in the future on the channels that would not be sourced from Java™ Messaging System (JMS) code segments.

Another interesting observation, after the fact – the architects had anticipated that one of the primary tenants of application deconstruction to services (be it rewritten or wrapped) was that business logic would not be tightly held within the application, thereby negating the specific business knowledge required by the development teams over time . It was anticipated that movement of the business rules to WebSphere Business Integrator (workflows) would, over time, offload IT from this space. In actuality, the business process modeling envisioned being done by the business is still an IT activity, with no signs of that ever moving. All that occurred was the shift of the process knowledge to another IT group (those charged with work flow creation). An acceptable outcome, as it still allows for strategies to increase global resource in the development areas, while keeping process-specific items in-house. As the tools to create automated services, BPEL etc. mature, the required skills in this arena will center around new IT roles, business architects and analysts.

During initial deployment, the team maintained a set of services that were only called by the internal MDI workflows and were not exposed for supplier interaction. Over time however, the lesson has been that exposing all of the services allows for additional flexibility and less work load back to IBM. The workflows were based on supplier process models, so it follows that if suppliers made process adjustments, that IBM's hidden services would require change.

Another major area underestimated during the initial project was the level of resource and new skill required to roll out this level of integration, not in development or planning, but in integration, test and deployment. Analysis shows that the message bus infrastructure is the main driver of complexity. Further, additional levels of SOA infusion are required to offset this effect in key back-end systems. To this end, IBM's roadmaps have been adjusted to address this area.

As this case study was in peer review, one of the reviewers, Karl Hancock, who manages a team of developers that actually write the services used on the MDI had the following comments which I thought were a very good addition to essons learned and a very valid set of reminders:

- Regardless of the services model, at integration time the devil is in the details, which still requires old fashioned, roll up your sleeves, get down and dirty, IT work effort, including process definition, transaction specifications, data modeling, and testing, testing, testing (among other things) i.e. expect significant effort and workload with first time deployments.
- The business has to make conscious trade-offs between control of a supplier' processes, and the degree of inter-dependency of one's processes with the supplier's processes. The services architecture doesn't change that. It may make the control points more specific and quantifiable, but a services solution can get undesirably intertwined in a supplier's processes regardless.

- The services model's effectiveness is limited by the effectiveness of the business process that employs the services. Business processes without BP owners, and without detailed definition, including process and result metrics, will flounder, with or without a services model underneath them."

"What is the current roadmap?". At present the main theme is simplification, the drive to remove the complexity within the integration structures highlighted above. First, categorize the suppliers. Our MDI level of integration is not the right fit for all, and for many suppliers the use of existing B2B is sufficient. Also, multiple MDIs per supplier are not required; the risk of single point of failure has proven to be offset by very high Mean Time To Failure (MTTF) and key nodes retrofit to High Availability infrastructure. As such, the present strategy is one MDI per supplier, that makes use of their internal connectivity to manage additional sites.

To address the complexity of the messaging bus driving large integration testing efforts and deployment resources, there is a pilot of a central ESB that will bring well known interfaces via services to a common point. The testing then ends there, with backend systems no longer required to participate directly on existing services. It also frees development groups from single dates where all groups can be ready, allowing for more agile development which returns productivity to development units at a better rate and pace. In addition to an ESB investigation, the division will be migrating its WebSphere Message Broker to WebSphere Process Server®1. Use of the process server will allow standardization of business process modeling efforts. The output (in BPEL, Business Process Execution Language) can then be delivered direct to WPS for a more timely response to business direction changes.

Business Benefits

This project demonstrated that SOA can be used for mission-critical development and that it is resilient and sound as a foundation. In addition, the reuse is real; an internal study for SOA metrics conducted by the IBM CIO office shows that in the 4th year of SOA investment. IBM GES division, on average, reuses 60% of services in new projects. Said another way, each year and half on average, the development teams are gaining the equivalent of a developer – virtually. This is significant when multiplied by the many IT service teams.

Conclusion

The thought "Do not try this at home", probably has some generic merit here. Overall, the success of the Virtual manufacturing project resulted from a highly skilled team, a team dedicated to success and accustomed to working on extreme, deadline driven projects.

The IT team capitalized on a growth enablement opportunity for the business to engage SOA as a methodology and the respective tooling associated and inventions required to achieve the goals in short order. There is no doubt that success would have been achieved with more conventional means, but not in the timeframe required with the scalability requested without adding additional resources. To this end, a final item is required – management support. IBM GES management understood the risks involved, the potential unknowns and chose to engage and stay the course. This is a very important point – without the support of management and stakeholders, SOA projects do not succeed.

Ford eHub

After early success in using SOAP interfaces in business-to-business applications, Ford enterprise architects, plant floor systems architects, and application teams in Europe and North America collaborated to apply SOAP, XML and other emerging Web Services standards to the problem of application interoperability, both internal and external to the enterprise. The result was "eHub", a standards-based real-time interoperability solution. It used XML over HTTP, digital signature, and OAGIS.

Ford started using the eHub in production for integrating plant applications with enterprise and supplier-facing applications in 2001. The initial implementation in 2001 supported two production manufacturing applications that made use of an event-driven real-time reliable messaging style of web services, as shown in Figure 26. A pilot was also done in material planning that exploited the eHub's capability for managing workflows that involve systems and human actors. The entire project to create this capability, from inception to production, took nine months.

The initial architecture deployed in 2001 was robust, and subsequent versions added more capability for high availability. This architecture provided capability for plant to business communication that previously was not available:

1. Scalability: XML flows over HTTP from plants to the data center. XML flows over HTTP just as efficiently as HTML (actually, more efficiently in general, since XML doesn't bring embedded graphics with it). Ford controls the timing and volume of the XML transmissions. Ford can decide what has to be real-time event-driven, what to trickle in packages every half hour. If desired, Ford can use FTP to send large files in batch mode. The web application on the B2B Server then accesses the database in the data center, which gives best user response times.

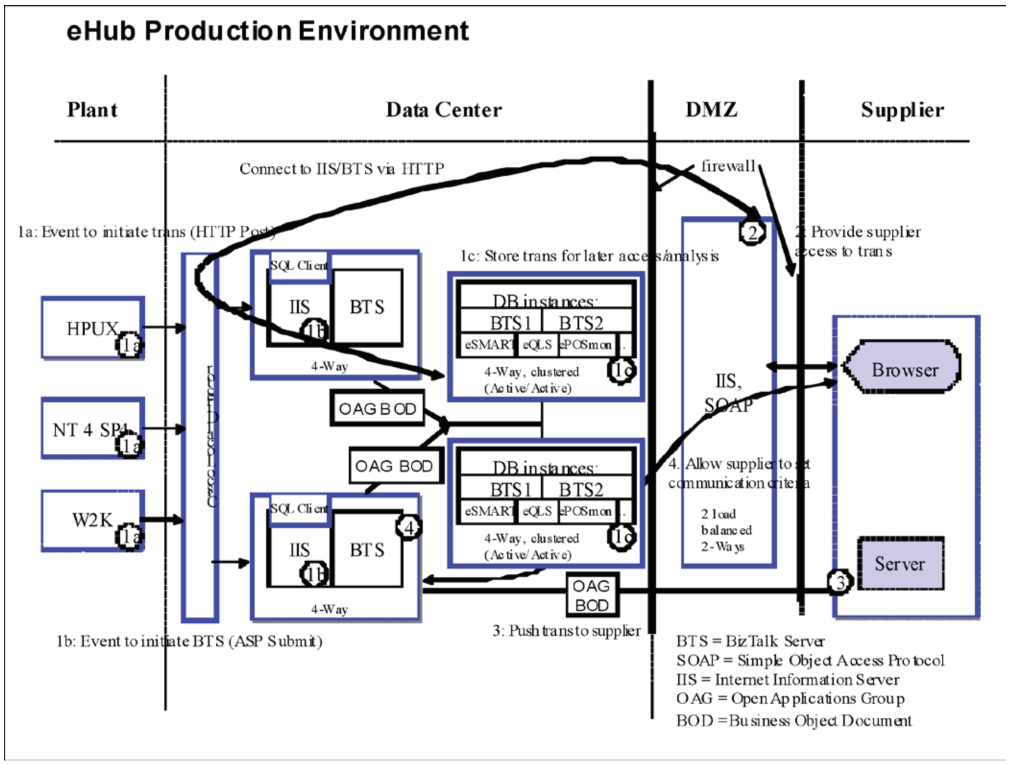
2. Data from all plants is in one database, so the arrangement supports multi-plant reports and ad hoc queries with the usual tools.

These are the OAGIS BODs that the eHub production implementations have used for manufacturing since 2001:

"Show Consumption" BOD in Scenario 60: "Vendor Managed Inventory (Consumption)"

"Show WIP Status" BOD in Scenario 57: "Production to Manufacturing Execution"

Figure 26: Ford eHub's Initial 2001 Production Environment


eHub Production Environment
Plant
Data Center
DMZ
Supplier
Connect to IIS/BTS via HTTP
firewall
1a: Event to initiate trans (HTTP Post)
2. Provide supplier access to trans
1c: Store trans for later access/analysis
SQL Client
IIS
BTS
4-Way
HPUX
NT 4 SP4
W2K
DB instances:
BTS1
BTS2
eSMART
eQLS
ePOSmon
4-Way, clustered (Active/Active)
OAG BOD
IIS, SOAP
Browser
4. Allow supplier to set communication criteria
2 load balanced 2-Ways
Server
OAG BOD
3: Push trans to supplier
1b: Event to initiate BTS (ASP Submit)
BTS = BizTalk Server
SOAP = Simple Object Access Protocol
IIS = Internet Information Server
OAG = Open Applications Group
BOD = Business Object Document

The value of eHub is that applications are integrated using a predicable, repeatable development process and the integration code is reusable. Also, because of the open standard technologies that are used, Ford's trading partners and software providers can easily and inexpensively connect their systems.

To implement a B2B or P2B application, developers use components in the eHub infrastructure that support the following process:

1) Choose appropriate standard XML, or invent new, consistent with standards (use Ford XML standards process, which starts with OAG, STEP and other market centric XML content standards). Ford's Enterprise Architecture group includes a small team of XML architects who assist projects with this step and maintain Ford's XML standards.

2) Build capability in the application(s) to emit the chosen XML message. Initially this was done by hand. Even when done manually, this was inexpensive to do. Later Ford's J2EE Center of Excellence developed standard Java code that any application team could use to connect to the eHub.

3) If the information goes directly to the user, allow the recipient to choose how to receive it, including details such as: when and how often, by e-mail, to a pager, wireless device, browser, an application listening at a URL, etc.

4) If the scenario initiates a dialog with another application, use a standard dialog (from RosettaNet, OAGIS, and more recently new ones that have been developed by the Automotive Industry Action Group).

Steps 3) and 4) were well supported by the infrastructure without much additional development.

True to its name, eHub has a hub and spoke architecture. For that reason, and because it was initially deployed to support production-critical manufacturing applications, it was designed to be very robust. That enabled it to be successful, although there were a few initial difficulties.

Soon after the initial implementation of the "Show WIP Status" BOD for communication with out-bound logistics providers, this information flow was re-used for communication with a system that controlled release of vehicles from the plant. Normally, this communication would be implemented in such a way that it should stay within the plant. Because the eHub communication was re-used, however, communication went from the plant (in Canada, Mexico, various parts of the U. S.) to the enterprise data center in Detroit, and then back. On the whole, the WAN was robust, but there were one or two instances in which WAN connectivity was lost. As a result, concerns were raised about the use of eHub. A project was proposed and technical design was done to replace the eHub connection with a direct connection within the plant, but this was never implemented.

The lesson learned is that any hub and spoke architecture has a single point of failure, and will be blamed for problems even if the messaging infrastructure itself is not the true root cause of the problem (although the probability of a hub failure is greatly reduced through highly available infrastructure solutions).

In 2002 Ford's Enterprise Architecture group installed an updated, higher capacity version of the eHub, the manufacturing applications were migrated to it, and the servers used for the initial installation were re-purposed. In 2003, just two years after its initial deployment eHub was handling about 5 million messages a month, with capacity for much more. It supported over 20 projects to integrate applications for Human Resources, Marketing & Sales, Product Development, Purchasing, and Manufacturing. It integrated Ford Motor Company with suppliers, dealers, and logistics providers, and was used in North America, Europe and Asia.

Usage of the eHub has continued to expand since then. By 2006 the eHub was used by at least 24 plants in North America and Europe. Nine trucking companies received vehicle status information in real time from the 18 assembly plants in Canada, Mexico and United States. The eHub is just one example of Ford Motor Company's commitment to excellence and innovation.

Appendix B: White Paper Authors and Reviewers

Authors

Alan Boyd
IBM Corporation
561-862-2774
alboyd@us.ibm.com

David Noller
IBM Corporation
866-405-7060
nollerd@us.ibm.com

Paul Peters
IBM Corporation
877-760-8247
pdpeter@us.ibm.com

Dave Salkeld
IBM Corporation
919-882-6110
salkeld@us.ibm.com

Tim Thomasma
Capgemini
734-730-9112
Tim.Thomasma@capgemini.com

Charlie Gifford
21st Century Manufacturing Solutions LLC
208-788-5434
charlie.gifford@cox.net

Steven Pike
IBM Corporation
802-769-3424
srpike@us.ibm.com

Alison Smith
AMR Research, Inc.
617-574-5213
ASmith@amrresearch.com

Reviewers

George Hudson
IBM

Bas Pluim
IBM

Aaron LaBella
IBM

Julie Fraser
Industry Directions Inc.

Paul Ashmore
Independent MES Consultant

David R Hinkler
Rockwell Automation

Mohammed Zuhair
Rockwell Automation

Alicia Bowers
GE Fanuc

Kay Freund
IBM

Victor Valle
IBM

Appendix C: Glossary / Acronyms

Name	Definition
A2A	Application to Application
AIAG	Automotive Industry Action Group
APS	Advanced Planning and Scheduling
Artifact	???Run-time element???
B2B	Business to Business
B2MML	Business to Manufacturing Markup Language
BATCHML	Batch Markup Language
BI	Business Intelligence
BOD	Business Object Definition
BPEL	Business Process Execution Language
BPM	Business Process Management
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
Canonical	According to standard and accepted rules, fitting the standard, and able to express full meaning simply
CAPA	Corrective Action and Preventative Action
CBE	Common Based Event – now standardized by OASIS as part of the WSDM (Web Services Distributed Management) standard: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
CID	Change in Design
CMM	Coordinate Measuring Machine
CMMS	Computer Maintenance Management System
CORBA	Common
COTS	Commercial off-the-shelf
CRM	Customer Relationship Management
DCS	Distributed Control System
DTD	Document Type Definition
DMZ	Demilitarized Zone – portion of network behind a firewall containing computers with IP addresses accessible to the internet.
ECE	Enterprise Composition Environment
ECO	Engineering Change Order
eHub	Name of integration hub in Ford integration example
EDMS	Enterprise Document Management System
EDW	Enterprise Data Warehouse
EIMS	Enterprise Integration Messaging Specification
EJB	Enterprise JavaBean(s)
Enums	Refers to “user defined enumerated types” in a programming language such as C++.
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
EWI	Electronic Work Instructions
FTP	File Transfer Protocol

GUI	Graphical User Interface
IEC	International Engineering Consortium
IIOP	Internet Inter-ORB Protocol
IMS	Integration Messaging Specification
ISA	Instrumentation, Systems, and Automation Society. Standards-setting organization for manufacturing operations management and automation
ISA-88	Standard Specification for batch processing industries
ISA-95	Enterprise – Control System Integration specification ISA-dS95.01-1999, ISA, July, 1999.
J2EE	Java 2 Platform, Enterprise Edition
JDBC	Java Database Connectivity
JMS	Java Message Service
LIMS	Laboratory Information Management System
MCE	Manufacturing Composition Environment
MDI	Multi-Data Source Integrator (name of hub in IBM example)
MDM	Master Data Management
Mfg MDM	Manufacturing Master Data Management
Mfg Ops	Manufacturing Operations
MES	Manufacturing Execution System
MIMOSA	Machine Information Management Open Systems Alliance
MOM	Manufacturing Operations Management (ISA level 3 application set)
MSB	Manufacturing Service Bus (ESB extended with manufacturing content such as services and integration standards support)
MTTF	Mean time to failure
NPI	New Product Introduction
OAG	Open Applications Group
OAGi	Open Applications Group, Inc
OAGIS	Open Applications Group Integration Specification
OI	Operations Intelligence
OPC	Open Linking & Embedding (OLE) for Process Control
ODS	Operations Data Store
OPM	Operations Process Management
P2B	Plant To Business
P&PLM	Product Process Lifecycle Management
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
QOS	Quality of Service
RFID	Radio Frequency IDentification
RPC	Remote Procedure Call
SCA	Service Component Architecture
SCADA	Supervisory Control and Data Acquisition
SCM	Supply Chain Management
SDO	Service Data Objects
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol

SPC	Statistical Process Control
SSL	Secure Sockets Layer
TBG	Trade and Process Business Group
TCIOps	Tool Control Infrastructure Operations
TLS	Transport Level Security
TREAD	Transportation Recall Enhancement, Accountability and Documentation Act of the US Government in automotive
UDDI	Universal Description, Discovery and Integration
UI	User Interface
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
UPS	Uninterruptible Power Supply
VMF	Virtual Manufacturing Framework
VMI	Vendor-managed inventory
VoIP	Voice Over Internet Protocol
W3C	World Wide Web Consortium
WBF	(used to be) World Batch Forum
WCF	Windows Communication Foundation
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
WSDM	Web Services Distributed Management
WS-I	Web Services Interoperability
WSRR	WebSphere Service Registry and Repository
XML	eXtensible Markup Language
XPATH	XML Path Language
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Translation

Appendix D: Trademarks

DB2 is a registered trademark of International Business Machines Corporation.

IEC is a registered trademark of International Engineering Consortium.

ISA is a trademark of The Instrumentation, Systems and Automation Society.

Microsoft is a registered trademark of Microsoft Corporation.

MQ Series is a registered trademark of International Business Machines Corporation.

OAGIS is a registered trademark of the Open Applications Group, Incorporated.

OPC is a trademark of OPC Foundation.

RosettaNet is a registered trademark of the RosettaNet Consortium.

Tivoli is a registered trademark of International Business Machines Corporation.

WebSphere is a registered trademark of International Business Machines Corporation.

WebSphere Process Server is a registered trademark of International Business Machines Corporation.

Windows is a registered trademark of Microsoft Corporation.

Appendix E: References

1. OAGi White Paper – Plug and Play Business Software Integration, The Compelling Value of the Open Applications Group, David Connelly, CEO, Open Applications Group, Inc, 1995.
2. The Open Applications Group Integration Specification, Michael Rowell, Open Applications Group, Inc, June, 2003.
3. XML Excellence at Ford Motor Company, Tim Thomasma, XML Journal, Volume 3, Issue 9, September, 2002.
4. OAGIS 8: Practical integration meets XML Schema, Mark Feblowitz, XML Journal, Volume 3, Issue 9, September, 2002.
5 XML in the Auto Industry: Summer 2002, Pat Snack and Sig Handleman, XML Journal, Volume 3, Issue 9, September, 2002.
6. ANSI/ISA-88.01, Batch Control Part 1: Models and Terminology, ISA, 2000.
7. ANSI/ISA-88.00.02, Batch control - Part 2: Data Structures and Guidelines for Languages, ISA 2001.
8. ANSI/ISA-95.00.01, Enterprise-Control System Integration Part 1: Models and Terminology, ISA 2000.
9. ANSI/ISA 95.00.02, Enterprise-Control System Integration Part 2: Object Model Attributes, ISA 2000.
10. ISA-88, Batch Control- Part 3: General and Site Recipe Models and Representation, ISA 2002.
11. ANSI/ISA-95.00.03, Enterprise-Control System Integration Part 3: Models of Manufacturing Operations Management, 2005.
12. ISA-88, Batch Control - Part 4: Batch Production Records, ISA 2006.
13. ISA-95 Part 5 (Draft 10), Business-to-Manufacturing (B2M) Transactions, ISA 2006.
14. WebSphere Service Registry and Repository Handbook, ISBN: 0738489972
15. Standards for Manufacturing Systems Integration, ISA-95 and OAGIS White Paper Series, White Paper #2: OAGIS, ISA-95 and Related Manufacturing Integration Standards, A Survey, ISA/MESA Publication, 2/1/2007
16. Standards for Manufacturing Systems Integration, ISA-95 and OAGIS White Paper Series, White Paper #1: An Overview and Comparison, ISA/MESA Publication, 2/1/2007

Appendix F: Partial List of Datatypes and Mappings in J2EE and .NET

Table 2: Partial List of Datatypes and Mappings in J2EE and .NET

Simple Type	Java Type	.NET Type
xsd:string	java.lang.String	String
xsd:integer	java.math.BigInteger	Int64 ?
xsd:int	Int	Int32
xsd:long	Long	Int64
xsd:short	Short	Int16
xsd:decimal	java.math.BigDecimal	Decimal
xsd:float	float	Single
xsd:double	double	Double
xsd:boolean	boolean	Boolean
xsd:byte	byte	SByte
xsd:QName	javax.xml.namespace.QName	String ?
xsd:dateTime	java.util.Calendar	DateTime
xsd:base64Binary	byte[]	Byte(Array)
xsd:hexBinary	byte[]	Byte(Array)
xsd:time	java.util.Calendar	DateTime ?
anyURI	java.lang.String	System.Uri ?
anySimpleType	java.lang.String	String ?
xsd:date	java.util.Calendar	DateTime ?
xsd:negativeInteger	java.math.BigInteger	System.Decimal
xsd: nonNegativeInteger	java.math.BigInteger	System.Decimal
xsd: nonPositiveInteger	java.math.BigInteger	System.Decimal
xsd:unsignedInt		UInt32
xsd:positiveInteger	java.math.BigInteger	System.Decimal ?
xsd:unsignedLong	java.math.BigInteger	UInt32/UInt64 ?

About IBM Corporation: IBM is the world's largest information technology company, with 80 years of leadership in helping businesses innovate. Drawing on resources from across IBM and key IBM Business Partners, IBM offers a wide range of services, solutions and technologies that enable customers, large and small, to take full advantage of the new era of e-business.

About Capgemini: Capgemini, one of the world's foremost providers of consulting, technology and outsourcing services, enables its clients to transform and perform through technologies. Capgemini provides its clients with insights and capabilities that boost their freedom to achieve superior results through a unique way of working – the Collaborative Business Experience – and through a global delivery model called Rightshore®, which aims to offer the right resources in the right location at competitive cost. Present in 36 countries, Capgemini reported 2007 global revenues of EUR 8.7 billion (approximately US$12 billion) and employs over 83,000 people worldwide.

About MESA: MESA promotes the exchange of best practices, strategies and innovation in managing manufacturing operations and in achieving plant-floor execution excellence. MESA's industry events, symposiums, and publications help manufacturers, systems integrators and vendors achieve manufacturing leadership by deploying practical solutions that combine information, business, manufacturing and supply chain processes and technologies. Visit us online at http://www.mesa.org.